ON SIMULATION METHODOLOGY IN VEHICULAR TRAFFIC FLOW

A THESIS

Presented to

The Faculty of the Division of Graduate

Studies and Research

By

Michael Pierce Deisenroth

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

in the School of Industrial and Systems Engineering

Georgia Institute of Technology

March, 1974

ON SIMULATION METHODOLOGY IN VEHICULAR TRAFFIC FLOW

Approved:

D. C. Montgomery

R. G. Heikes

R. S. Jones

P. H. Wright

D. B. Young

Date approved by Chairman: May 8, 1974

ACKNOWLEDGMENTS

I would like to express my appreciation to Dr. D. C. Montgomery who served as my advisor throughout this research. He provided the guidance and encouragement which was necessary for successful completion of this work.

I am also grateful to the other members of the reading committee, Dr. R. G. Heikes, Dr. P. S. Jones, Dr. P. H. Wright and Dr. D. B. Young. Their suggestions and observations have greatly improved this dissertation.

Dr. R. N. Lehrer, Director of the School of Industrial and Systems Engineering, has provided leadership and support. My special thanks go to him for his help and understanding.

I also wish to thank Mrs. Betty Sims. Her excellent typing has contributed to the quality of both of my theses.

My most special thanks go to my wife Claudia for her patience, encouragement, endurance and faith. I am greatly indebted to her and our two daughters, Meg and Michele, for their sacrifice and support.

This work is dedicated to my students. May I show them the patience and understanding that others have shown me.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

LIST OF ILLUSTRATIONS

Figure Page

SUMMARY

Earlier work in modeling traffic flow of the sort treated here has established that analytical models are too limited but that the general technique of stochastic event simulation is fruitful. Several traffic simulation methodologies have evolved. This research compares and contrasts the methodologies in order to provide insight into future areas of model research and development.

Four computer models, representing different strategies for simulating urban vehicular traffic flow, were programmed and tested. Webster's experimental model was used to validate each simulation model for an isolated intersection.

A series of tests were conducted to determine the significance of model type, network configuration and flow rate. A statistically significant difference in the delay predictions of the models was found. Although the time-dependent characteristics of delay predictions obtained from the various models appeared to follow the same pattern, the magnitudes of delay predicted by the models differed significantly.

Additional tests were run on selected models to investigate some secondary considerations. The scan time interval selected was found to significantly affect delay predictions. Although the results regarding model preloading were not conclusive, evidence indicated that network preloading should result in reduced computation time for large congested systems. The correlation between delays observed in different time

periods appeared to be restricted to a one-cycle lag. However, more research is warranted before results in this area can be used to indicate simulation running time.

CHAPTER I

INTRODUCTION

The design or improvement of traffic control systems depends on a model that simulates the relevant features of the vehicular flow and the controlling mechanism. Earlier work in modeling traffic flow of the sort treated here has established that analytical models are too limited but that the general technique of stochastic event simulation is fruitful. Several traffic simulation methodologies have evolved. Their piecemeal development and reporting in the literature tend to make their distinctions and interrelationships obscure, and it is this situation that creates the need for the research reported in the present thesis.

Purpose of Research

The purpose of this research is to compare and contrast existing methodologies in the simulation of traffic flow. It is hoped that these studies will provide a means of evaluating the relative merits of these different strategies. By identifying the strengths and weaknesses of the different techniques, insight into future areas of model research and development should be realized.

Limitations and Scope

The research is limited to models of traffic flow in urban vehicular networks with specific emphasis on models used to evaluate traffic control alternatives. Specifically excluded from this research

is the work done in the areas of open road simulation and freeway design. Although the fundamental modeling techniques are similar, different system configurations and vehicular behavior characteristics are considered. (Freeway and open road models typically have longer roadway segments and special intersection geometry where passing and merging movements are considered.)

Four different models, representing basic modeling strategies, are compared with respect to three different traffic situations--the isolated intersection, an urban arterial flow and a medium-sized closed network. Each situation is further stratified to test the effects of traffic volumes on model performance. Light, medium and heavy service volumes are considered for each system configuration.

Objectives

The primary objective of this research is the identification, isolation, comparison and evaluation of different simulation methodologies as applied to traffic flow. Consideration is given to model size, computation speed, flexibility and response prediction. Both dynamic and static behavior of the models are compared.

As a secondary objective, three tactical problems of traffic simulation were investigated. First, different models and situations are tested to evaluate model initialization and warm-up. Preloading of the roadway is investigated for two of the models considered. Next the time-scan interval of two models is varied to test the sensitivity of simulation response and computation time to this parameter. Finally,

the dynamic output of the models is investigated to analyze its effect on total simulation running time.

CHAPTER II

LITERATURE SURVEY

In the last two decades, digital simulation of vehicular traffic flow has attracted considerable interest among traffic flow theorists and practitioners. A review of all the literature that has been published in this area is beyond the scope of this study. Appended to this thesis is a bibliography of relevant literature which represents the majority of available literature in this area. This review will concentrate on outlining the general areas of traffic simulation and the different simulation approaches which have been employed.

Simulation Situations

Traffic flow simulation can be divided into three basic areas: intersection models, network simulation and freeway or highway models.

Intersection Simulation

Considerable effort has been devoted since 1956 to the simulation of the individual intersection. The work of Goode, Pollmar and Wright (18)* represents the pioneering contribution. A single four-legged intersection with a fixed-time traffic signal was modeled. Turning movements were permitted and left-turning vehicles checked the approaching traffic for an acceptable gap before turning. The roadway was

---

*Numbers enclosed in parentheses indicate references listed in the Bibliography of Cited Literature.

represented by a string of binary digits indicating the presence or absence of a vehicle. Every quarter second the vehicles were examined to see if they could be advanced.

Next, two separate studies of intersection simulation were performed by Benhard (2) and Lewis (27) in 1959. These studies dealt with the simulation of an intersection of two two-lane streets with actuated signal control. The models were greatly simplified, however, in that turning and passing were prohibited.

Lewis continued his work in intersection simulation with the development of a second model (28). This model was developed to determine volume warrants for intersection control. Two types of intersection control were studied: the two-way stop sign and the semi-actuated signal. The model permitted both turning and passing maneuvers and parking was accommodated on the minor street. The individual approaches were represented by a one-dimensional coordinate system and vehicles were permitted to assume any desired position along the approach. Extensive car-following rules were developed for use with the model. A scan time of one second was employed.

Kell (23,24) constructed models to study intersection delay for two-way stop, fixed-time and vehicle-actuated intersections. The traffic volumes were varied and the resulting delays were compared for each control method. Additional runs were made at selected volumes to determine the effect of turning maneuvers on intersection delay. By concentrating on time relationships among the vehicles, an event scan program was developed which boasted a 7000:1 ratio of real time to simulation time.

Gerlough and Wagner (16) developed a model to study traffic performance at an isolated intersection. A continuous spacing representation similar to Lewis's was adopted. Many driver characteristics were represented by probability distributions. A vehicle's following behavior was dictated by a reciprocal spacing model. In this model acceleration (deceleration) was proportional to relative speed and inversely proportional to spacing. A scan time between 0.25 and 0.5 seconds was permitted.

After having performed extensive field work to obtain relevant distributions regarding the models, Thomasson (36) and Wright (43) simulated stop sign controlled intersections, two-way and four-way, respectively. Specific emphasis was placed on driver reaction time and lag and gap acceptance.

Apparently the latest published work in the area of intersection simulation as of early 1974 was reported by Barnes (1) in 1971. The primary goal of his research was to formulate general models of seven basic types of intersections ranging from a four-way stop model to a four-lane, two-lane signalized model with two turn lanes. These models were written in GPSS, a general purpose simulation language.

Network Simulation

As soon as initial efforts were completed on simple intersection simulations, research workers began to consider the problems of simulating a portion of an urban transportation network. Goode and True (19) combined four single-intersection models to form the first network model. An oscilloscope was used to diplay the movement of vehicles through the network.

In 1961, Stark (35) developed a model for a nine-block section on 13th Street, N.W., Washington, D. C. The model represented a combination of signal-controlled and stop-sign-controlled intersections. Additionally, vehicles were allowed to pass as well as accelerate and decelerate. A total of 37 main routines, subroutines and table look-up routines were included in the model. The roadway was divided into 12-foot unit blocks. Positioning could be specified to the nearest 1/100th of a block. The roadway was updated every quarter of a second.

Katz (22) and Gerlough and Wagner (13) reported the initial development of TRANS, the first general network simulator, in 1963. TRANS was written to evaluate the effects of different traffic signal settings in some specified region of a large network. The initial experiment was on a section of Washington, D. C., which contained 80 signalized intersections. A macroscopic view of vehicle behavior was adopted. The simulation scan interval indicated was five seconds. Traffic lanes were divided into zones. The length of a zone was such that a car moving at free speed moved from one zone to the next in one time period.

Francis and Lott (11) have reported another model developed for simulating a general network. This program was applied to a series of nine traffic signals along a main traffic route in Central London. Although general in structure, the model was not capable of handling large networks such as the Washington, D. C. application.

During that same year, a third network model was reported by Rhee (32). Although few details of this model are available, it seems quite

similar to the TRANS model.

GPSS has found numerous applications in network simulation. Blum (4) proposed that intersection modules be used as generalized building blocks for the formation of any network configuration. This initial work was expanded (5) until now it is a general purpose network model capable of useful application to practical problems. In 1966, Schwartz (34) utilized GPSS II in programming a general purpose network model, which was applied to a segment of downtown Boston. Voskoglov and Wheeler (37) also made use of GPSS to study the effects of proposed parking facilities adjacent to the University of Missouri campus. Although this model represented a network application, the model developed was not a general-purpose model.

Sakai and Nagao (33) produced a model to simulate large networks. This program was field tested on a small area of Kyoto, Japan. A traffic display board was constructed for this experiment to provide a visual means of validating the model. The model divided the roadway into 50-meter blocks capable of containing a number of vehicles. The number of vehicles advanced was related to a transfer function and depended on relative density. A scan interval of four seconds was utilized.

The latest network model is reported by Lieberman, Worrall and Bruggeman (29) and is called UTCS-1. It is a microscopic simulation model designed to evaluate urban traffic control systems. The capabilities of the model include the ability to model vehicle-pedestrian conflicts, bus route and bus stop action, and intersection spill-back.

Freeway and Highway Simulation

Concurrent with the development of intersection and network models was the investigation of freeway traffic flow. Gerlough (12) reported the first application of simulation to freeway traffic. He studied a quarter-mile section of one-way, two-lane traffic. Each car had a preferred velocity, which was maintained when possible. Cars traveling below their desired speed were permitted to pass slower vehicles. As in earlier intersection work, a series of binary digits indicated positioning along the roadway.

The Midwest Research Institute (17) utilized simulation to study the interchange design problem. Actual data from four Chicago interchanges were used. Gap acceptance decisions were determined by distributions empirically derived. The freeway was divided into a series of blocks 17 feet long. During each second of simulation time, basic rules of advancement were checked to determine the new position of the vehicle.

Wohl (41) discussed the freeway merger problem as an illustrative example of simulation application. His basic model differs considerably from that of the Midwest Research Institute in that it was more concerned with time spacing than with actual position.

Howat (21) developed a model to investigate the driver-automobile interactions with respect to overtaking, following and passing on the open road. A decision table was used for modeling driver behavior. In 1966, Cassel (8) also studied overtaking and passing on the open road. Drivers were permitted to abort a passing attempt and accidents could

occur if the automobile could not respond quickly enough to new conditions.

Buhr, Besserole and Drew (7) reported the simulation of a section of freeway with multiple on and off ramps. The model was general in nature and could be applied to different situations. The overall structure of the program is similar to that of the Midwest Research Institute. However, of particular interest in the study was the use of a pre-loaded roadway.

Modeling Philosophies

Inherent in the simulation models developed to date and in the four models to be described in the next chapter are different philosophies for performing the actual simulation procedure. It is the purpose of this thesis to identify these philosophies and to establish their effects on model performance, speed and size. Before describing the models that were studied, different methods of delineation will be reviewed.

Physical Representation Versus Memorandum Representation

Gerlough (14) was one of the first researchers to attempt to classify different modeling approaches. He divided the then existing models into two general categories--physical representation and memorandum representation.

With the physical representation approach, one or more binary digits are assigned to represent static location blocks in a roadway. The data contained in these blocks can represent presence, position and possibly size of a vehicle on a specific segment of roadway. Areas of

computer memory are organized to represent the network being studied. Vehicles are advanced by manipulating the data in the static location blocks.

The memorandum method focuses attention on the individual vehicles. The characteristics and status of the vehicle form the basic unit of the data structure. Vehicles are advanced by changing the values of specific status variables such as velocity and position according to some prescribed logic.

Time Scanning Methodologies

For the purposes of this thesis, a more useful delineation can be obtained by expanding the work of Kell (26). Kell emphasized the importance of the time scanning methodology by first classifying the models as either fixed time scanning or next event scanning. Then he divided the fixed time scanning models into models which utilized a discrete roadway representation and those employing a continuous roadway representation.

With the fixed time scan methodology, the state of the system being simulated is examined at regular intervals of time. Vehicle position and motion characteristics are then updated and the process repeated at the next point in time. The actual position of the vehicle may have a discrete representation, causing the vehicle to move along the roadway by jumping from one point to another, or it may closely approximate a continuous representation.

The event scan methodology relates the actions of vehicles to specific points in time. Examples of these events are system entrance,

queue joining, intersection entrance and turn completion. Since each event considered in the scanning procedure has some known or determinable position associated with it, the sequence of events determines the sequence of vehicle positions. Position, therefore, does not need to be defined, and the event time of occurrence describes the necessary space/time relationships of vehicles.

The models described in this thesis can be classified according to Kell's scheme. Three fixed time scanning models are described which utilize different methods of representing the vehicle/roadway relationship. The final model is characteristic of the event scanning methodology.

CHAPTER III

DESCRIPTION OF THE MODELS

Common Features of the Models

A number of features were common to most, if not all, of the models developed in this research. A description of these features will precede detailed discussions of the individual models.

The General Form of the Network

Since it was desired to be able to simulate a variety of different situations, a general form of network representation was adopted. In all four models a traffic network is considered to be a set of *links* and *nodes*. A node represents a traffic intersection, a traffic source or a traffic sink. A link is a unidirectional road segment between two network nodes. Therefore, a one-way street between two intersections would be regarded as a single link while a pair of links would be necessary to represent a two-way connecting street. Figure 1 shows an illustrative network consisting of 12 nodes and 24 links.

Network nodes are classified as either external nodes or internal nodes. The external nodes act as traffic sources and sinks and may not represent actual network intersections. In Figure 1 nodes 1 through 8 are external nodes. The internal nodes represent intersections in the real network and are exemplified by nodes 9 through 12. For the purposes of this study, all internal nodes are assumed to be signalized with a fixed time signal controller. Signal parameters, such as cycle

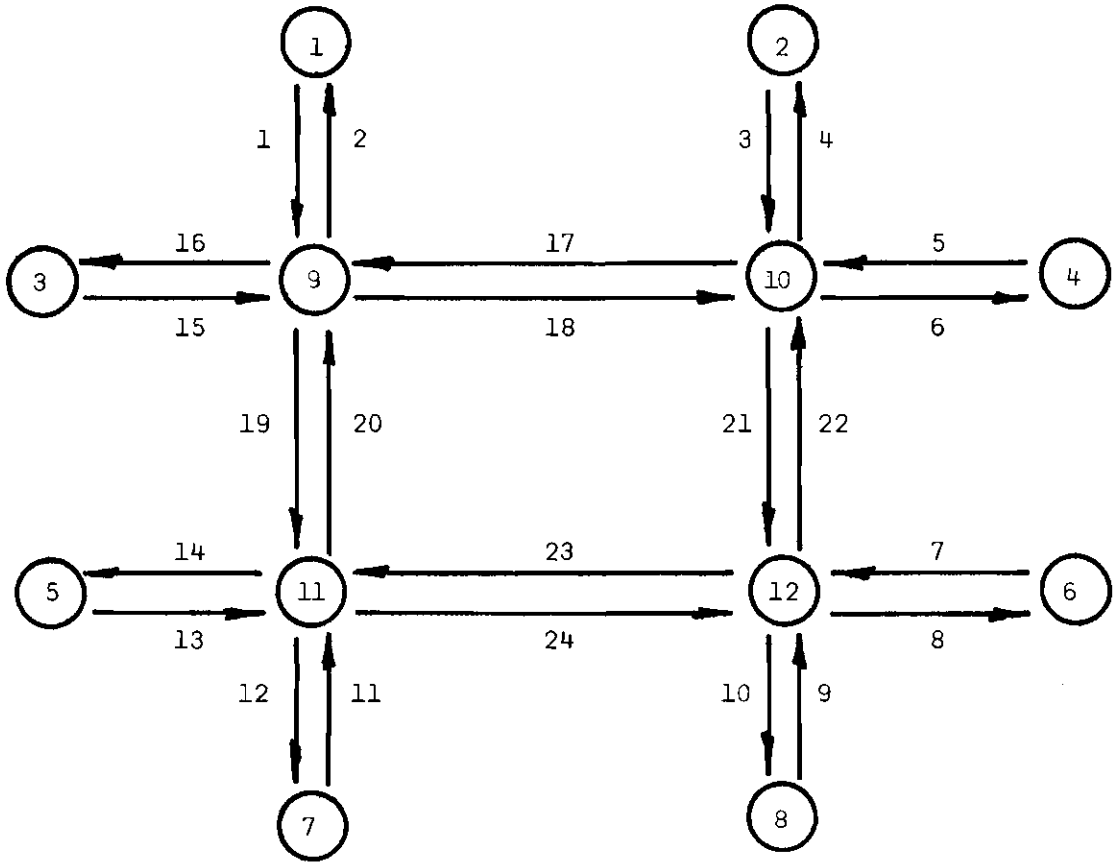


Figure 1. Illustrative Traffic Network

length, green time, and offset, are required input data for each signal.

Each link is identified by stating the node number associated with it or by a unique index. For example, in Figure 1, link 24 goes from node 11, the tail or source node, to node 12, the head node. The index for the head node is also the signal index. The characteristics of each link must be supplied as input data. These include length, width (in lanes) and the turn probabilities associated with the head node.

To simplify matters somewhat and to provide consistency between the different models, a standard input format was adopted for all four models and is described in detail in Appendix A. Comments regarding assumptions and restrictions on network characteristics will be made throughout the remainder of this thesis.

Pseudo-Random Number Generation

The pseudo-random number generation method used is known as the multiplicative congruential method. The general procedure is described as follows. Let

$$X_n = A \cdot X_{n-1} \pmod{m},$$

where A is a constant multiplier,

m is the modulus, and

X_o is the initial random number seed.

The sequence, $\{X_n\}$, approximates a sequence of integers uniformly distributed between zero and the modulus. By dividing X_n by the modulus,

a sequence $\{R_n\}$ of pseudo-random numbers is obtained.

Vehicle Generation

The time between arrivals of vehicles at an input node is assumed to be an exponentially distributed random variable. This distribution has been found to be fairly satisfactory at low volumes, but is usually inaccurate at moderate and high volumes. The significance of this assumption is examined in Chapter IV.

A separate random number sequence is used for each generation node. One additional random number sequence is used for all other Monte Carlo techniques. To insure that the volumes generated by each input node were within acceptable limits, the vehicle generation segment was isolated from the programs and five hours of generation were conducted at volumes of 200, 300, 400, 500, and 600 vehicles per hour. The number of arrivals per five-minute time period were recorded and tested against a Poisson distribution. Chapter IV summarizes the results for the selected random number seeds.

Turning Behavior

The destination of a vehicle on a given link is determined by a Monte Carlo technique. The probabilities of right turns, straight-through movements and left turns are specified as input for each link in the network. Three of the models, which treat vehicles individually, assign a desired movement to each vehicle as it enters the link. This desired movement is then used to select an appropriate lane on multi-lane links. The fourth model assigns the turning movement when the vehicle reaches the intersection.

Speed Distribution

The desired velocity of an individual vehicle is determined by a Monte Carlo technique for those models which treat vehicles individually. When the vehicle is created, it is assigned a desired velocity which remains fixed throughout its journey in the network.

By using a distribution of desired velocities, these models introduce a source of delay which is not found in models where a single desired velocity is assumed for all vehicles. Vehicles which are not affected by the intersection control device or by preceding turning vehicles may still be delayed by the need to slow behind a vehicle with a lower desired velocity. This source of delay depends not on the intersection control scheme but on the standard deviation of the desired velocity and the length of an individual link.

The exclusion of this distribution was thought to detract from the flexibility and realism of the models and it was therefore included where possible. To minimize the resulting differences in delay prediction, all models which included a speed distribution were forced to assume the same form. A slightly skewed representation of a normal distribution was used for all models that permitted a variable desired velocity.

Left Turn Gap Acceptance

The gap acceptance procedure utilized by all four models is a Monte Carlo technique for assessing gaps in the opposing flow. A random number is compared to the probability of accepting the existing gap. If the probability value is greater than the random number, the gap is

accepted. The distribution used is indicated by Figure 2 and is an approximation of distributions reported in the literature (36). No distinction is made between a gap and a lag.

Measures of Effectiveness

The models include as the principal part of their output certain measures of effectiveness that were thought to be important in evaluating system performance. The primary measures of effectiveness are:

System Delay

Queue Lengths

Number of Stops

The definition and method of obtaining data on each of these measures will be deferred to the individual model presentations. The differences in interpretation is felt to be a major contributor to lack of consistency between the models (26).

The Continuous Model

The most detailed traffic simulation models have approximated a position of a vehicle by a continuous variable. The position of the vehicle is changed by using laws of rectilinear motion and the current state of the system. A continuous roadway representation introduces little rounding error due to positioning; however, rounding error due to time scanning still exists.

Lewis (28) and Gerlough and Wagner (16) utilized a continuous representation for modeling a single intersection while Buhr, et al. (7) chose the continuous approach to model freeway exit and entrance ramps. For these studies a detailed description of vehicle interaction was

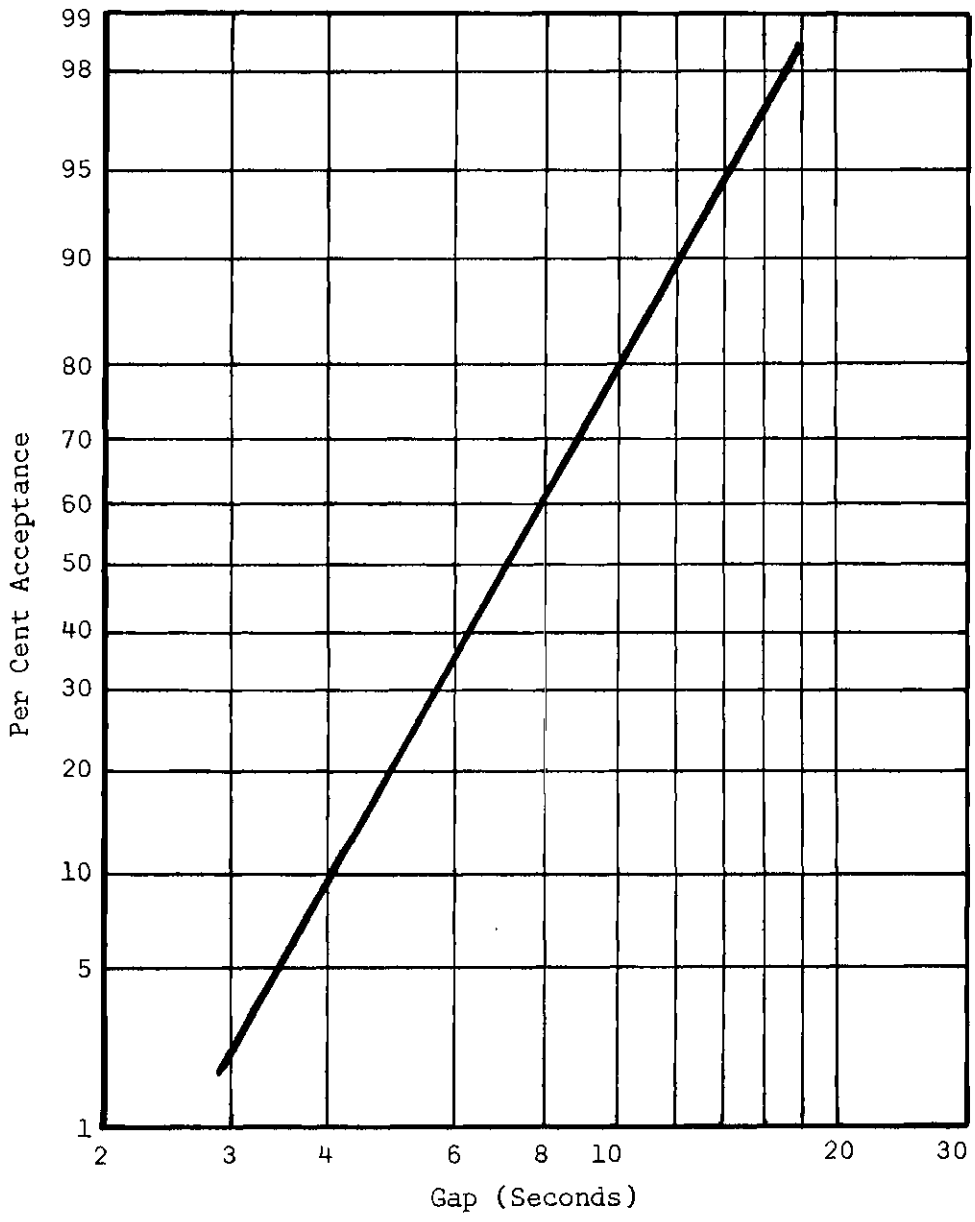


Figure 2. Gap Acceptance Distribution

thought necessary to measure vehicular delay and to obtain realistic acceleration and deceleration procedures. The continuous model employed in this study will be referred to as the CONT model and draws heavily from the model developed by Lewis.

Roadway Representation

In the CONT model, each lane is a one-dimensional coordinate system. The link entrance is assumed to be the projection of the curb line at the tail of the link and is designated as the zero coordinate point. Link length is the distance, in feet, from this point to the projection of the near side curb line at the head node. The stop line is located 12 feet back from the extension of the curb line.

The coordinate system for each lane is extended into the intersection, since vehicles do not exit one link until they pass the entrance point of the next link. The coordinates of the end points are dependent on the intersection geometry and turning movement desired. In Figure 3 the end points of a one-lane and two-lane approach are illustrated. The complete lane stationing is included in Table 1.

Table 1. Intersection End Points

Number of Lanes and Movement	Distance to End Point from Near Side Curb Lane (Feet)
One-Lane Approach	
Left Turn	40
Straight Movement	30
Right Turn	12
Two-Lane Approach	
Left Turn	85
Straight Movement	60
Right Turn	12

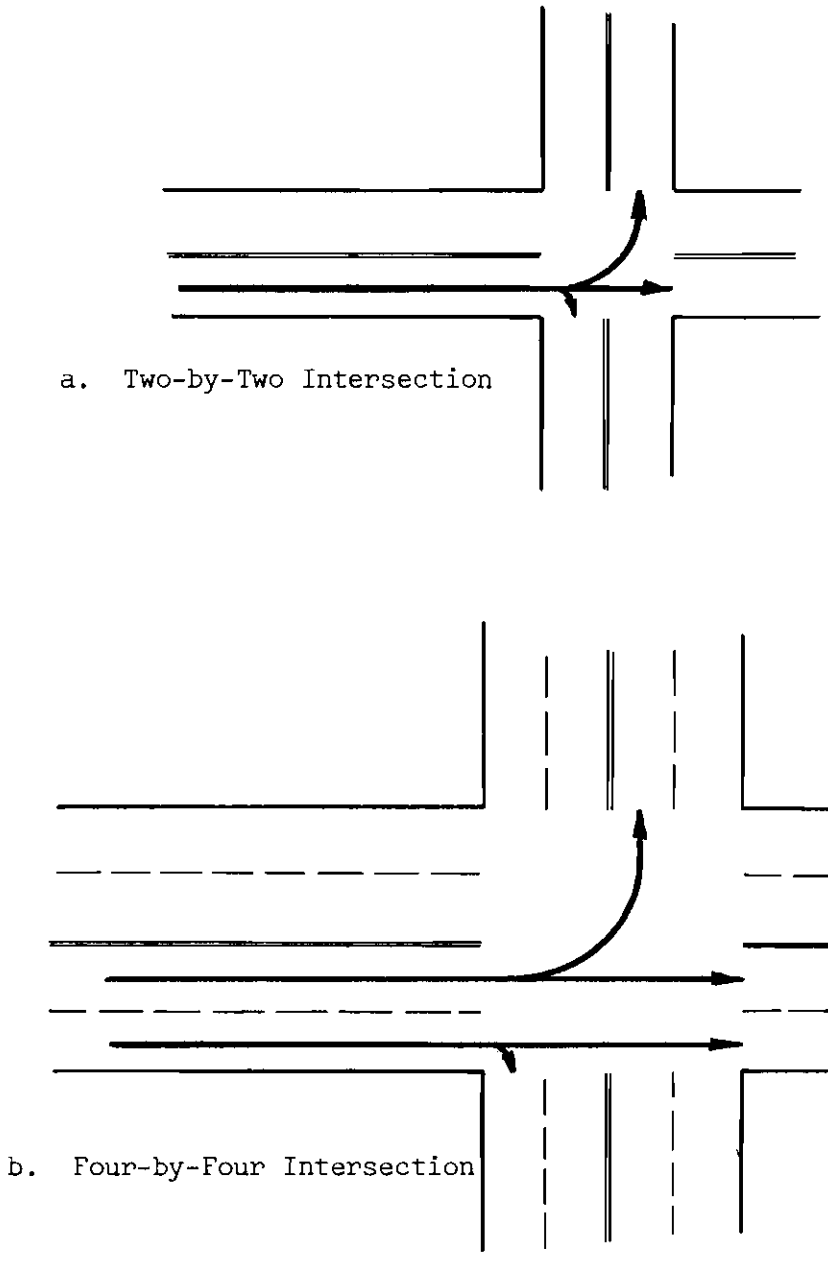


Figure 3. Link Endpoints for Two Types of Intersections

Because end points are a function of intersection geometry, the program must contain data on all possible intersection configurations to be included in the network. Since only two-lane by two-lane and four-lane by four-lane intersections were included in this study, the table is quite small. Expansion of the table would not be a major task if additional intersection geometries were included in the network.

Vehicle Representation

The vehicle in the CONT model is represented by a set of 12 computer words. These words contain data on the characteristics and status of the individual vehicle. Specific data are maintained on:

1. Desired Speed
2. Present Link
3. Present Lane
4. Desired Turning Movement
5. Estimated Link Exit Time
6. Position
7. Actual Speed
8. Index of the Vehicle in Front
9. Index of the Vehicle in Back
10. Stopping Tag
11. Acceleration Tag
12. Holding Movement Distance

As indicated earlier, the desired speed is a fixed characteristic of the vehicle and remains constant throughout the travel of the vehicle in the network. Items 2 through 5 are assigned values as the vehicle enters a new link. Items 6 and 7 are updated every time scan period based on the assumed motion of the vehicle during that time interval. A one second scan period was adopted. Items 8 and 9 provide a means of establishing the ordering of the vehicles on a given link. For each lane of each link a record identifying the first vehicle and the last vehicle is maintained. By starting with the first vehicle on a specific

link and examining the information contained in item 9, the second vehicle can be identified. This procedure can be repeated to determine the proper ordering of all vehicles on that link. Items 10 through 12 are associated with intersection movement.

Vehicle Movement

The heart of any continuous model is the car-following vehicle behavior relationships. Car-following theories attempt to relate the reaction of a following vehicle to the motion of a leading vehicle.

One form of the spacing-speed relationship assumes that spacing is a linear function of speed. Lewis used this relationship to structure his model behavior. A second form of car-following behavior proposes that the response of a following driver is the product of a sensitivity factor and a stimulus. Chandler, Herman and Montrol (9) suggest that the stimulus be the relative velocity of the two vehicles and that a response of acceleration or deceleration be delayed some constant reaction time. Gerlough and Wagner (16) used this function with the addition of an inverse relationship to vehicle spacing. The CONT model is a generalization of the linear space/speed relations used by Lewis. A detailed development of the equations to follow can be found in Reference 28.

Acceleration Restriction. It is assumed that all vehicles attempt to travel at some desired speed, which is a characteristic of each vehicle. A uniform rate of speed change is assumed under free flow conditions. A vehicle, otherwise unrestricted, will accelerate to its desired velocity at an acceleration rate $\bar{A}$ of 4 ft/sec^2. Let ZA be the distance

that a vehicle travels in one second based on this restriction. Then

$$ZA = \min\{V_{t-1}+0.5\bar{A}, 0.5(V_{t-1}+V_{max})\},$$

where V_{t-1} is the velocity of the vehicle one second ago,

V_{max} is the maximum desired velocity for the vehicle and,

$\bar{A}$ is the desired constant acceleration rate.

Once at the desired velocity, the vehicle will maintain that velocity until some other restriction is encountered.

Car-Following Restriction. When the spacing between two vehicles is critical, the speed of the second vehicle is adjusted so that at the end of a time scan the new position is no closer than desired. The desired spacing assumes that vehicles moving at the same speed will maintain a spacing linearly proportional to their velocity and a fast overtaking vehicle will keep sufficient distance to permit a safe stop.

Using these assumptions, Lewis has derived the maximum distance a vehicle can advance in one second based on the car-following restrictions. Let ZS be the maximum distance a vehicle can move in one second. Then when $V_{t-1} > V'_t$

$$ZS = 0.5(V_{t-1}+V'_t) - 0.75\bar{D} + [0.5625\bar{D}^2-0.25\bar{D}V_{t-1}+0.75V'_t+0.5(X'_t-X_{t-1}-P)]^{\frac{1}{2}},$$

and when $V_{t-1} \leq V'_t$

$$ZS = [X'_t-X_{t-1}-P+V_{t-1}]/3,$$

where

ZS is the maximum distance a vehicle can move in one second,

V_{t-1} is the velocity of the following vehicle one second ago,

V'_t is the present velocity of the lead vehicle,

X_{t-1} is the position of the following vehicle one second ago,

X'_t is the present position of the lead vehicle,

$\bar{D}$ is the desired deceleration rate, and

P is the assumed minimum spacing obtained at zero velocity.

Stopping Restriction. As a vehicle approaches an intersection, the traffic control device may restrict its movement. When a traffic signal turns from green to amber, the associated input links are "tagged." The first vehicle in each link which can stop within reasonable deceleration limits is then tagged and it begins decelerating to a complete stop. Preceding vehicles in that lane are allowed to continue through the intersection while those vehicles following the tagged vehicle will stop as a result of the car-following restriction. Only one vehicle per lane is tagged. When a traffic signal turns to green, all associated input tags are removed.

Let ZD be the maximum distance a vehicle operating under the stopping restriction may advance during a one-second period. Then

$$ZD = 0.5V_{t-1} - 0.25\bar{D} + [0.0625\bar{D}^2 - 0.25\bar{D}V_{t-1} + 0.5DX_{gap}]^{\frac{1}{2}},$$

where

ZD is the maximum distance under the stopping restriction,

V_{t-1} is the velocity of the vehicle one second ago,

$\bar{D}$ is the desired deceleration rate, and

X_{gap} is the distance from the vehicle to the stopping point.

If the intersection is blocked by a vehicle from a side street or by a left turning vehicle from the opposing link, a stopping restriction is applied to the first vehicle of each lane. The stopping point for these blockages is assumed to be one foot beyond the extension of the curb line.

Turning Restriction. During a turning maneuver it is assumed that a vehicle will decelerate to some desired speed then accelerate throughout the remainder of the movement. The point of inflection in the acceleration pattern is called the "turn point" and the location of the turn point is assumed to be a function of both intersection geometry and desired movement as indicated by Table 2.

Table 2. Intersection Turn Points

Number of Lanes and Movement	Distance from Near-Side Curb Line to Turn Point (Feet)
One-Lane Approach	
Left Turn	7
Right Turn	1
Two-Lane Approach	
Left Turn	13
Right Turn	1

Let ZT be the maximum distance which a vehicle can travel in one second under the turning restriction. Then

$$ZT = 0.5V_{t-1} - 0.25\bar{D} + [0.0625\bar{D}^2+0.25(V_{TP}^2-\bar{D}V_{t-1})+0.5\bar{D}X_{gap}]^{\frac{1}{2}},$$

where V_{t-1} is the velocity of the vehicle one second ago,

$\bar{D}$ is the desired deceleration rate,

V_{TP} is the maximum permissible turn point velocity, and

X_{gap} is the distance between the vehicle and the turn point.

This equation is only applicable when the vehicle does not proceed beyond the turn point. When this occurs a new value for ZT must be computed which will consider the assumed deceleration and acceleration.

Left turning vehicles will not proceed past the turn point unless an acceptable gap is found in the opposing stream of traffic. Each lane of the opposing link is searched to find the position and velocity of the first vehicle. The time gaps are then calculated for each lane and the minimum gap used to make a decision. If the minimum gap is accepted the vehicle proceeds through the turn point; if rejected the vehicle immediately stops at the turn point to wait for an acceptable gap. The gap acceptance table is then checked every time scan until an acceptable gap is found. Left turning vehicles which have passed the turn point accelerate at 7, 6 and 5 ft/sec^2 for the first three seconds after an acceptable gap becomes available.

Measures of Effectiveness

The CONT model provides output on system delay, maximum queue lengths and number of stops. All statistics are printed by individual link, by intersection and by total system performance.

Delay is defined as the difference between actual travel time and

undelayed travel time. Since the model assigns each vehicle a desired velocity from a specified probability distribution, undelayed travel time must be calculated for each vehicle. Furthermore, a vehicle which has just entered a link may not, and most likely will not, start at coordinate zero. The actual starting location will depend on the distance the vehicle was from the end point of the previous link at the beginning of the scan time and how far it advanced during that scan cycle.

To calculate undelayed travel time, a single vehicle was permitted to traverse a link and the travel time was recorded. Link length, desired velocity and desired turning movement were varied in order to obtain travel time as a function of these factors. As each vehicle enters a new link, the initial position, link length, desired speed and desired turning movement are considered in calculating the undelayed exit time. Upon exiting the link, this undelayed exit time is subtracted from the actual exit time to obtain delay.

If a vehicle slows to a speed less than 4 ft/sec, it is assumed to stop and the stop is recorded. Lewis (28) provided a means of measuring stop delay but this was not incorporated in the CONT model.

The estimation of queue length was difficult in this model. The accordion effect often witnessed in heavy urban traffic can exist in this model. Hence, a vehicle far removed from the intersection may be stopped, while vehicles in front of this stopped vehicle are accelerating. Since it was desired to estimate the maximum queue lengths obtained, the length of a queue was recorded each time the signal at the head of the length turned green. A routine was written to scan each

lane and find the last stopped vehicle. All vehicles in front of this vehicle were considered to be in the queue, even if they were moving greater than 4 ft/sec.

General Model Flow

The general flow of the model is indicated by Figure 4. A complete listing of the program is given in Appendix B. The initialization phase, which is common to all four models, consists of the four subroutines:

EXTERN - to read data cards on external nodes.

INTERN - to read data cards on internal nodes and signals.

INPLKS - to read data cards on all network links.

SETUP - to manipulate data into desired form and schedule the first arrival for each input node.

The main body of the simulation consists of four primary subroutines, as indicated in Figure 4, and 12 secondary subprograms.

The SIGCHK subroutine checks the signal at each internal node to see if a change of state is desired. A change from green to amber will cause the appropriate input links to be tagged as described earlier.

The VEGEN subroutine checks each external node to see if it is time to create a new vehicle. If this is desired, a new vehicle is created and the next vehicle to enter is scheduled. The new vehicle is placed in the "holding area" of the appropriate link. This procedure will be explained shortly.

The LKMOVE subroutine is concerned with the movement of vehicles on the links and through the intersections. Each node in the network is checked individually. Input links for the node are examined one at a

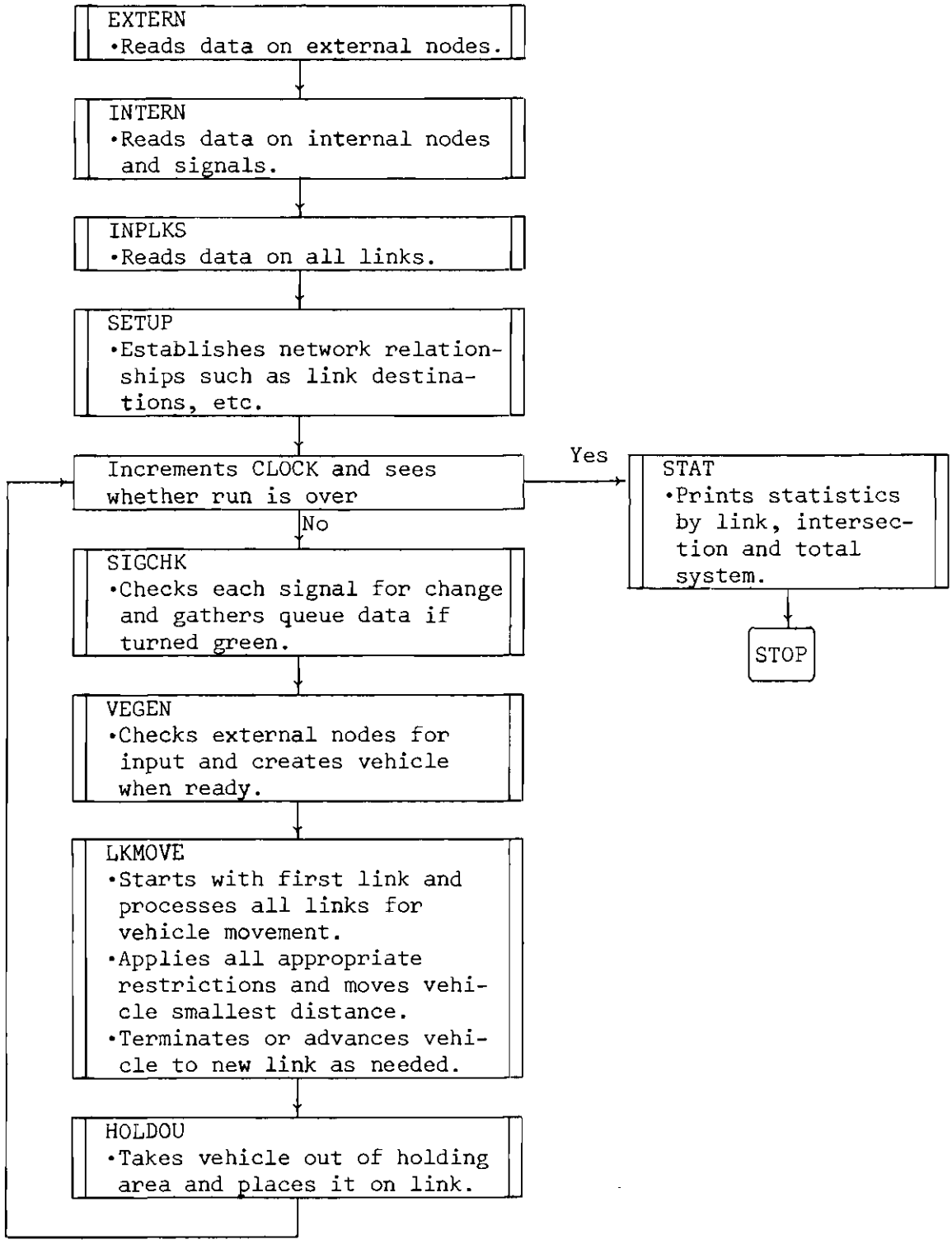


Figure 4. General Flow of CONT Model

time. Vehicles are processed starting with the first vehicle in the left-most lane and proceeding sequentially down the lane from the head to the tail.

As each vehicle is examined, the maximum possible distance it could advance is calculated for each of the four restrictions. The minimum of these distances is assumed to be the desired advance. If the desired distance will take the vehicle past the turn point (past the near side curb line for straight movement), the destination link is checked for available space before proceeding into the maneuver.

If the new position of the vehicle is beyond the end point of the link, it is removed from the link and placed in the "holding area" of the destination link. After all links have been processed, each holding area is checked for vehicles. These vehicles are then removed from the holding areas and allowed to start down their new links. This discontinuity of processing is necessary to avoid processing a vehicle twice in the same scan interval--once on the old link and then on the new link. Neither single intersection models nor arterial models present this problem, which results from the application of a general model to a closed loop network.

The purpose of the HOLDOU subroutine is to remove vehicles from the holding areas after all link movement is completed. Additionally, each vehicle is assigned a turning maneuver and an expected exit time.

The Unit Block Model

Many of the early efforts in traffic simulation relied on a discrete representation of vehicular positioning. Goode, Pollmar and

Wright (18) considered the roadway as a string of distinct points where vehicles were permitted to jump from point to point. Wong (42) proposed a slightly more general model by dividing the roadway into a series of blocks. A roadway segment one car length long and one lane wide was called a unit block. Each unit block could contain one car or it could be empty. Cars were confined to speeds of integral numbers of unit blocks per unit time.

Stark (35) developed a unit block model where blocks were 12 feet in length. The positioning of the vehicle within the block was to the nearest 1/100th of a block and vehicles could advance in the quarter second scan without changing blocks. The precision of Stark's model closely approximates that of a continuous model. The model developed for this study is patterned more closely to the Wong model than that of Stark or Goode et al. It will be referred to as the UB model.

Roadway Representation

The UB model divides the roadway into a series of 18-foot blocks, each one lane wide. The unit blocks are organized into an array in the computer memory. The presence or absence of a vehicle in a unit block can be determined by examining the value contained in the appropriate memory location. A zero implies that the block is empty while a positive number indicates the index of the vehicle occupying that specific roadway position.

The unit block notation provides a means for identifying and ordering the vehicles and specifying their location. Figure 5 illustrates two one-lane links connecting two intersections. By starting at the

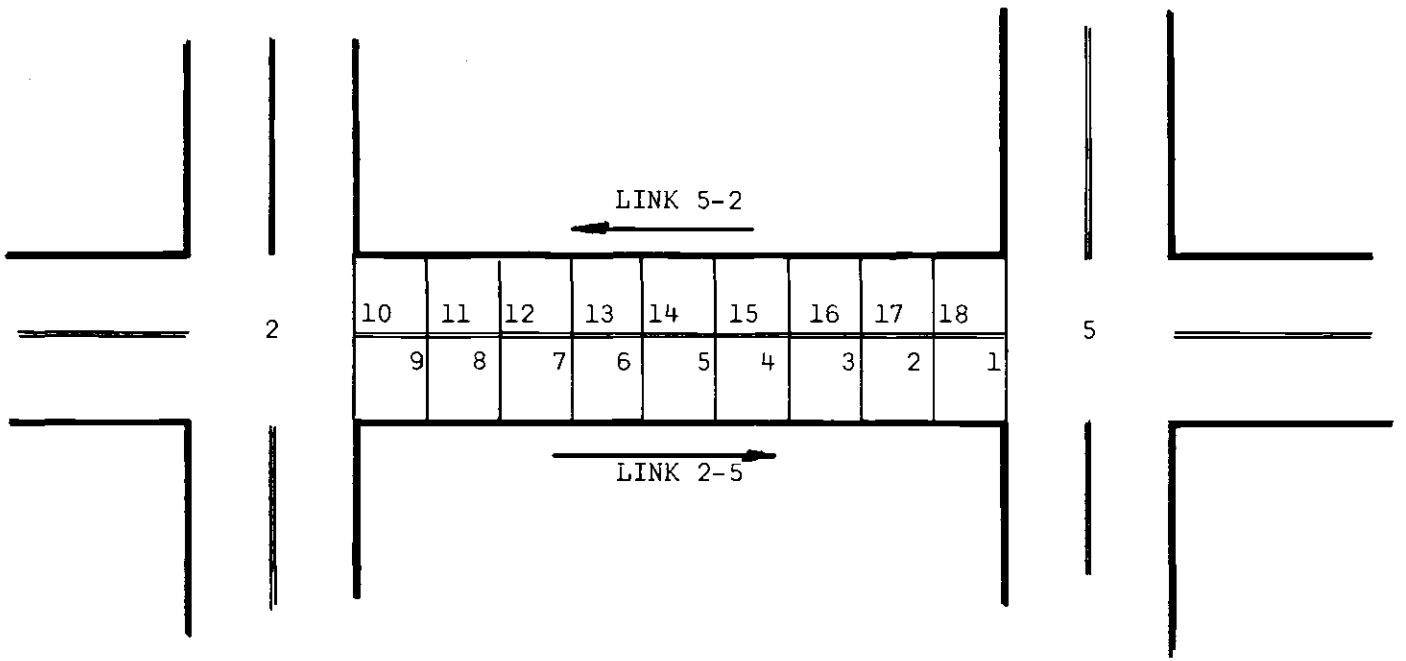


Figure 5. Unit Block Representation of a Two-Lane Road Segment

head of the link and working back to the tail, all vehicles on the link can be identified and ordered.

The representation of an intersection using a unit block approach requires additional logic. Stark (35) indicates that it was necessary to superimpose unit blocks from crossing streets. The UB model partitions the intersection into distinct intersection squares and requires a complex algorithm to associate unique computer memory locations with each square. This approach was taken to insure that two vehicles on different links were never, in reality, on the same segment of an intersection. Figure 6 identifies the numbering of intersection squares for a two-lane by two-lane and a four-lane by four-lane intersection. The Roman numerals identify the approach orientation for the given block numbering scheme.

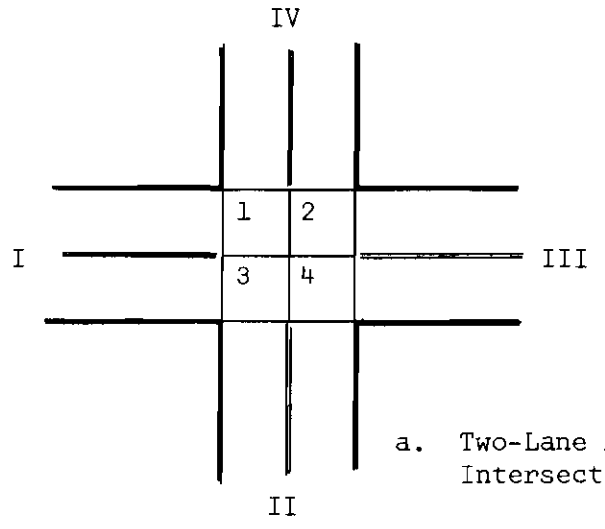
If additional intersection geometries were necessary for a specific application each unique geometry would require an explicit definition of the intersection partitioning.

Vehicle Representation

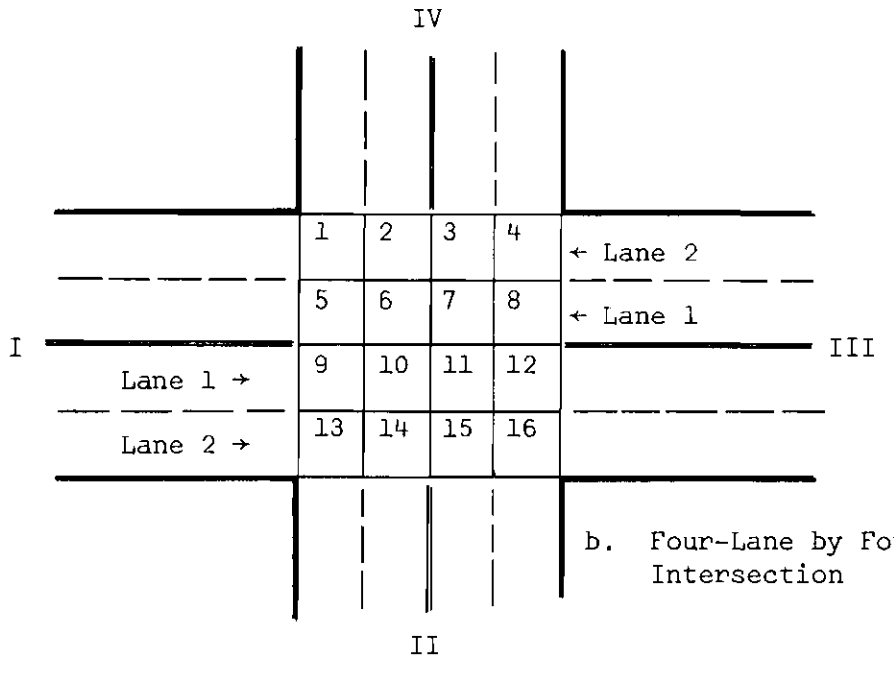
The vehicle in the UB model is represented by a set of nine computer words. The specific records maintained on each vehicle are:

1. Desired Speed (In Blocks Per Second)
2. Present Link
3. Present Lane
4. Desired Turning Movement
5. Estimated Link Exit Time
6. Actual Speed
7. Move Indicator
8. State Indicator
9. Holding Movement Distance

Since the desired speed is integral in block length, a limited number of speeds are possible. Assuming a block length of 18 feet, the



a. Two-Lane by Two-Lane Intersection



b. Four-Lane by Four-Lane Intersection

Figure 6. Intersection Diagrams for the UB Model

speed distribution assumed is indicated in Table 3. This same distribution was specified in those models which allowed variable desired speeds. It somewhat represents a normal distribution suggested by Gerlough and Wagner (16).

Table 3. Distribution of Desired Speeds

Desired Speed		Probability Percentage
Blocks/Sec	Feet/Sec	
2	36	7
3	54	90
4	72	3

Items 2 through 6 and item 9 have identical counterparts in the CONT model described earlier. The move indicator is set to one if the vehicle has been processed during the current scan; otherwise it is set equal to zero. The state indicator is a four state switch indicating whether the vehicle is accelerating, decelerating, traveling at a constant velocity or standing still.

Vehicle Movement

The complexity of the car-following equations found in the CONT model are avoided in the UB model; however the restrictions are very similar.

Intra-Link Movement. An otherwise unrestricted vehicle will accelerate to its desired velocity at the rate of 1 block/sec^2. This rate is considerably faster than the acceleration limit assumed in the CONT model. Therefore a restriction is imposed which limits

acceleration to every second scan cycle. This restriction reduces the average acceleration rate to 9 ft/sec^2.

After the new position based on the current velocity of the vehicle and the acceleration restriction has been calculated, the headway between this position and the next vehicle is examined. A table of minimum headways is referenced to insure that an adequate separation is being maintained. These minimum headways are a function of the relative velocities of the two vehicles as shown in Table 4. They represent a compromise between the headways found in the CONT model and the minimum safe following distances possible with a deceleration rate of one block per second.

Table 4. Minimum Acceptable Headway

Minimum Acceptable Headway					
Following Vehicle's Speed (Blocks/Sec)	Lead Vehicle's Speed (Blocks/Sec)				
	0	1	2	3	4
1	0	1	0	0	0
2	1	2	2	0	0
3	3	4	4	2	0
4	6	6	4	4	3

Vehicles desiring to turn must also check their new position with respect to the intersection. All turning vehicles are required to slow to one block per second velocity in the intersection area. A table look-up procedure establishes the maximum permissible turn velocities for various distances from the intersection.

All vehicles approaching an intersection with an amber or red light are required to stop. A table look-up procedure is employed to establish the maximum velocity possible based on the vehicle's current position.

If the advancement of the vehicle will carry it into the intersection, additional checking is required. A vehicle will not enter an intersection unless there is space available in the destination link. Furthermore, vehicles in the left-most lane will not enter the intersection if there is a left-turning vehicle in front. This restriction is necessary to insure that intersections will not become hopelessly jammed by opposing left-turn vehicles. Headway restrictions apply between vehicles in the intersection and vehicles that are attempting to enter the intersection.

Inter-Link Movement. Movement restrictions within an intersection depend on the turning maneuver desired and the intersection geometry. All turning vehicles are restricted to a 1 block/sec velocity for the first half of the maneuver. Once the turn is executed, a normal 1 block/sec^2 acceleration is assumed.

Figure 7 illustrates the paths followed by vehicles making different maneuvers. The sequence of intersection squares encountered is a function of direction of approach as well as intersection geometry and desired turning maneuver.

A left-turning vehicle advances two or three squares into the intersection before observing the gap in the opposing flow. Because of the discrete nature of velocities and locations the model gaps are only an

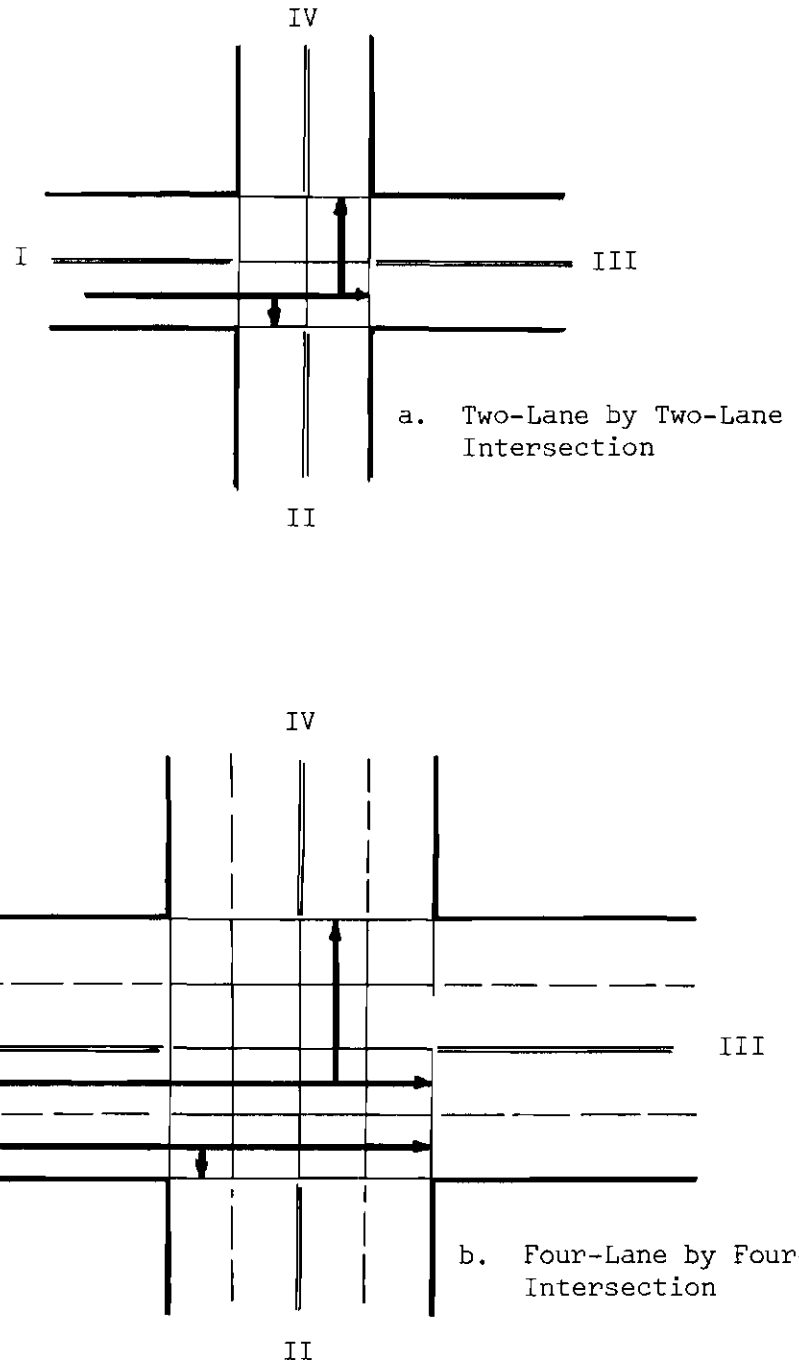


Figure 7. Intersection Maneuvers for the UB Model

approximation of the actual gaps. However, the error due to this approximation is not greater than .5 seconds, the intrinsic error caused by the nature of the model. If a left-turning vehicle rejects the gap and stops, the inter-link routine maintains an empty block behind the waiting vehicle to permit opposing left-turning vehicles to clear the intersection.

Measures of Effectiveness

The UB model provides output on system delay, maximum queue length and number of stops. All statistics are printed by individual link, by intersection and by total system performance.

Like the CONT model, the UB model calculates delay by subtracting the undelayed travel time from the actual travel time. Undelayed travel times were obtained by taking vehicles with different desired speeds through all combinations of turning maneuvers and intersection geometries.

A stop is defined as the act of going from a non-zero velocity to a velocity of zero. All stops are recorded by lane for each link in the network. Stop time delay is easily obtained from this type of model; however, it was not included in this study since it is not easily available with the other models included.

The queue length is evaluated for a link each time the light turns green. The largest queue found is assumed to be the maximum queue. To evaluate the length of the queue, each block, starting with the last block, is examined until a stopped vehicle is found. All vehicles in front of this vehicle are considered to be in the queue.

General Flow Model

The general flow of the model is illustrated in Figure 8. A complete listing of the FORTRAN program is given in Appendix C. The initialization phase consists of the four subroutines:

EXTERN - to read data cards on external nodes.

INTERN - to read data cards on internal nodes and signals.

INPLKS - to read data cards on all network links.

SETUP - to manipulate data into desired form and schedule the first arrival for each input node.

The main body of the simulation consists of five primary subroutines and 13 secondary subprograms. The basic structure of all three fixed time scanning models is very similar.

The SIGCHK subroutine checks the signals at each node to see if a change of state is desired. A change from red to green will cause the queue lengths to be examined on the appropriate inputs.

The VEGEN subroutine checks each external node to see if it is time to create a new vehicle. If this is desired a new vehicle is created and placed into the holding area of the appropriate link.

A separate subroutine was written to process inter-link movement for each intersection geometry considered in the network. The INTERL subroutine is a control routine which takes intersections one at a time and passes them to the appropriate processing routines. The MOVETT subroutine processes vehicles for a two-lane by two-lane intersection while the MOVEFF subroutine processes vehicles for a four-lane by four-lane intersection.

The intersection is first checked for vehicles from the non-green

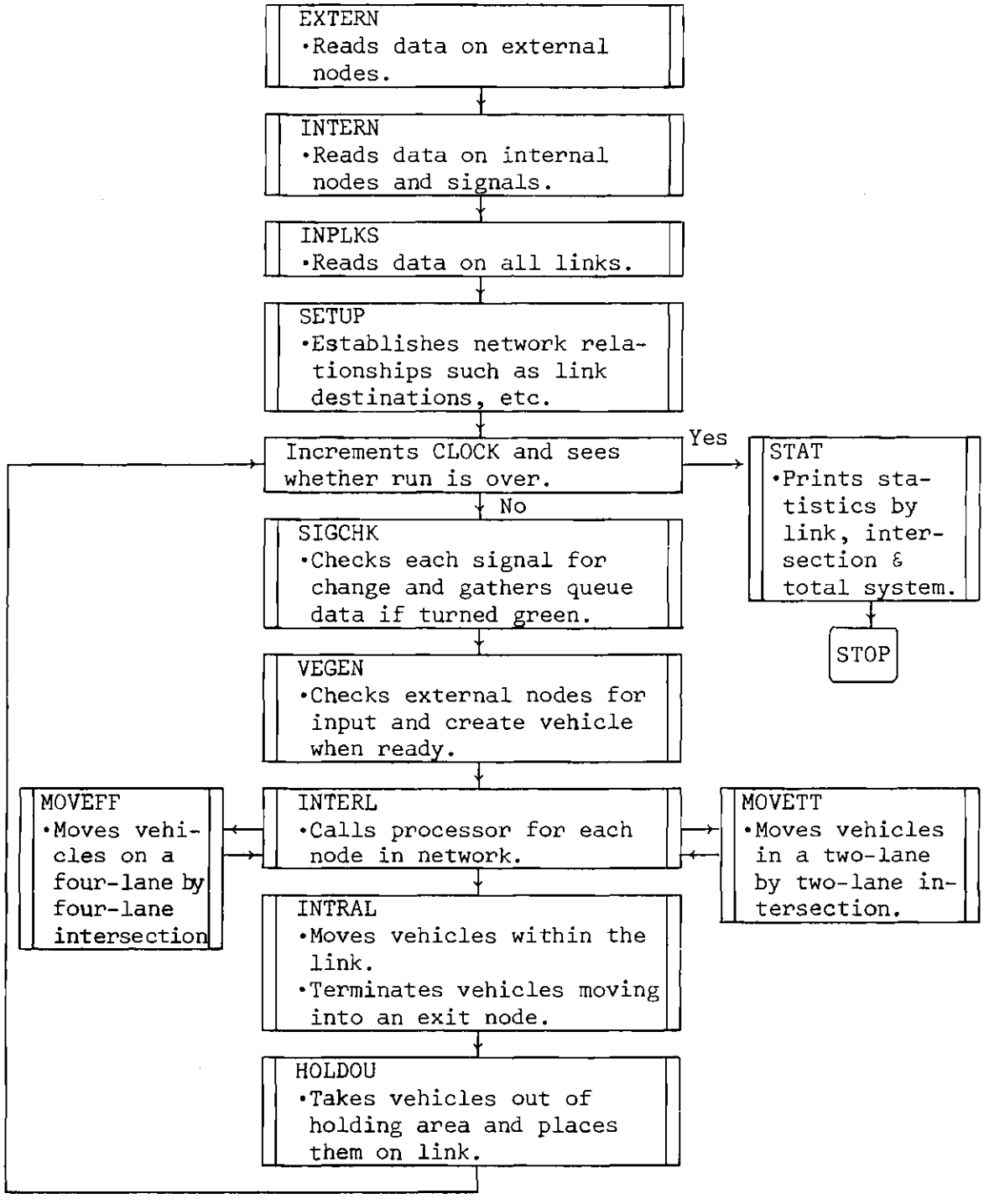


Figure 8. General Flow of UB Model

approaches. Then the green approach vehicles are considered for advancement. Within each approach scan, vehicles completing left or right turns are processed first. Then the lane extensions are scanned from destination to the start of the link.

If it were desired to model different intersections such as a two-lane by four-lane or an intersection with a one-way approach, a separate subroutine would be necessary for each unique configuration. A completely general network simulator using the concepts presented in the UB model would require a substantial amount of additional programming to apply to these different conditions.

The INTRAL subroutine processes vehicles traveling between two nodes. The scanning process begins with the first block at the head of the link. Each block on the link is inspected for the presence of a vehicle. If a vehicle is found, the movement restrictions are applied and the vehicle is advanced the desired number of blocks. All blocks are searched until the last block on the link is processed.

The HOLDOU subroutine removes vehicles in the holding areas and places them on the appropriate links. To avoid processing the same vehicle twice in the same scan, the HOLDOU subroutine is not executed until after all intra-link movements have been considered.

The Zone Model

The two models presented thus far have been microscopic in nature in that vehicles are individually maneuvered through the network with somewhat complex decision processes being simulated every time scan. In order to model a larger, more complex system of traffic, a macroscopic

approach has been adopted by a number of researchers.

Gerlough et al. (13,22) developed the TRANS model to evaluate the effect of traffic signal settings on traffic flow in a region of a city. Initially it was applied to a section of the District of Columbia containing over 300 links and 80 intersections. At about the same time Rhee (32) developed a program (Urban Traffic Control Simulator) which utilized a representation very similar to the approach taken by TRANS. Sakai and Nagao (33) adopted a network representation similar to TRANS, but the movement of vehicles was somewhat different and will be described below. The zone model developed for this research is patterned after the TRANS model and will be referred to simply as the ZONE model.

Roadway Representation

In the ZONE model each lane is divided into zones as indicated by Figure 9.

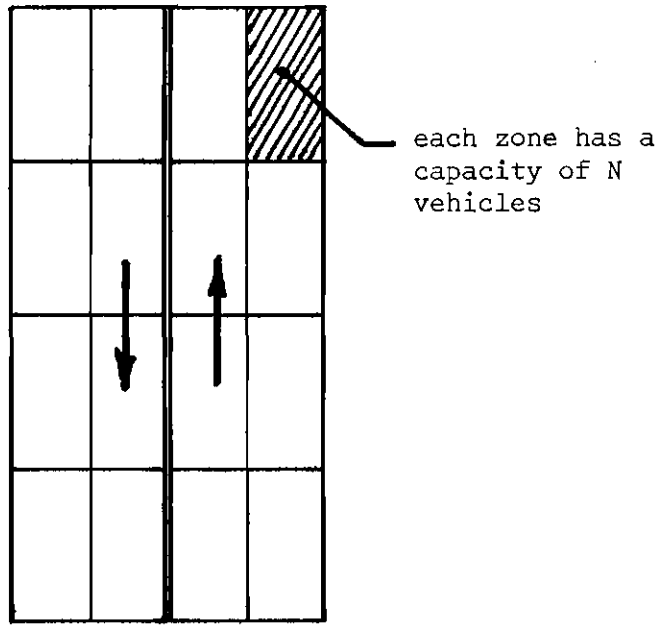


Figure 9. Two Two-Lane Links Divided into Zones

The length of a zone is such that a car moving at "free-flow" speeds from one zone to the next is the time interval between scans. In the initial version of TRANS a five-second scanning cycle was adopted to take full advantage of the macroscopic nature of the program. The ZONE model assumes the two-second scanning period which was reflected in more recent applications of TRANS (39).

A two-second scanning cycle and a free-flow speed of 53 ft/sec (the mean of the distribution used by the other models) implies a 106-foot zone length. Finally, assuming an average vehicle length of 20 feet, the capacity of each zone is five vehicles. To test the effects of a two-second scan cycle, a specific situation was tested under varying scan cycles, with results reported in Chapter IV.

The physical intersection has no counterpart in the model. Vehicles are transferred from the first (exit) zone of the source link to the destination link. This movement makes intersection geometry immaterial and, as a result, link geometry considerations (number of lanes, turn lanes, etc.) present the only restriction to network representation.

The ZONE model utilizes the computer words associated with individual network zones to indicate the number of vehicles on specific segments of roadway, whereas the UB model utilizes the block to indicate the index of the vehicle concerned.

Vehicle Representation

The ZONE model is unique among the four models tested in that vehicles are treated in clusters rather than individually. Only in the critical area of the intersection does the ZONE model resort to the

individual handling of vehicles. As a result of this no computer memory is allocated to storing data on vehicles.

Traffic Movement

The movement of traffic in the ZONE model is divided into inter-link movements and intra-link movements. Inter-link movements are characterized by a vehicle-by-vehicle consideration while intra-link movements manipulate platoons of vehicles.

Inter-Link Movements. The inter-link movements transfer vehicles from the first (exit) zone of the source link to the holding area of the destination link. The ZONE model has no representation of vehicles within an intersection. Vehicles are drawn from the exit zone one at a time. A Monte Carlo technique is applied to determine the destination link and the move is attempted.

An aborted exit will cause the model to stop processing the lane and proceed to the next lane of that link or a new link, whichever is appropriate. If a left turn fails to find an acceptable gap in the opposing flow, a flag is set and the model will not process any inter-link movements from that lane until the first vehicle completes a left-turn maneuver.

The number of vehicles which may be transferred from an exit zone depends on several factors. If the link is in the queue state, then a discharge rate restricts the number of vehicles which can be transferred. If the destination refused entry or if an acceptable gap is not found for a left turn, the transferring is terminated. Otherwise, all vehicles in the exit zone are assumed to move at the free-flow speed, 53 ft/sec, to

their desired destinations.

All vehicles in the exit zone are stopped by a red light. If the light is amber and if the turn flag indicates a vehicle is waiting for a left turn, that vehicle is permitted to traverse the intersection. All other vehicles are stopped by the amber light.

It should be pointed out that the method of roadway representation used in the ZONE model makes all gaps integral multiples of scanning time. For larger scanning cycle times, this assumption is much coarser than the acceptance graph indicates. For example, with a scan time of 5 seconds, only gaps of size 0, 5, 10, 15 and 20+ seconds are possible.

Intra-Link Movement. Once the first zone of a link has been processed, the remaining zones are processed by a very simple algorithm. The model advances as many vehicles from the $(j+1)th$ zone as can be accommodated in the jth zone. Any vehicles not advancing are considered to have been delayed.

Sakai and Nagao (33) proposed a more complex scheme for advancing vehicles from zone to zone. A transfer formula was specified which related the number of vehicles advanced to the relative speed of the two zones concerned. The zone speed was estimated based on block density. Figure 10 indicates the propagation of a traffic stream under this formula. Although this approach seems desirable, the approach employed in the TRANS model was incorporated in the ZONE model. This approach allows queue discharge restrictions to spread out a platoon and are computationally more efficient.

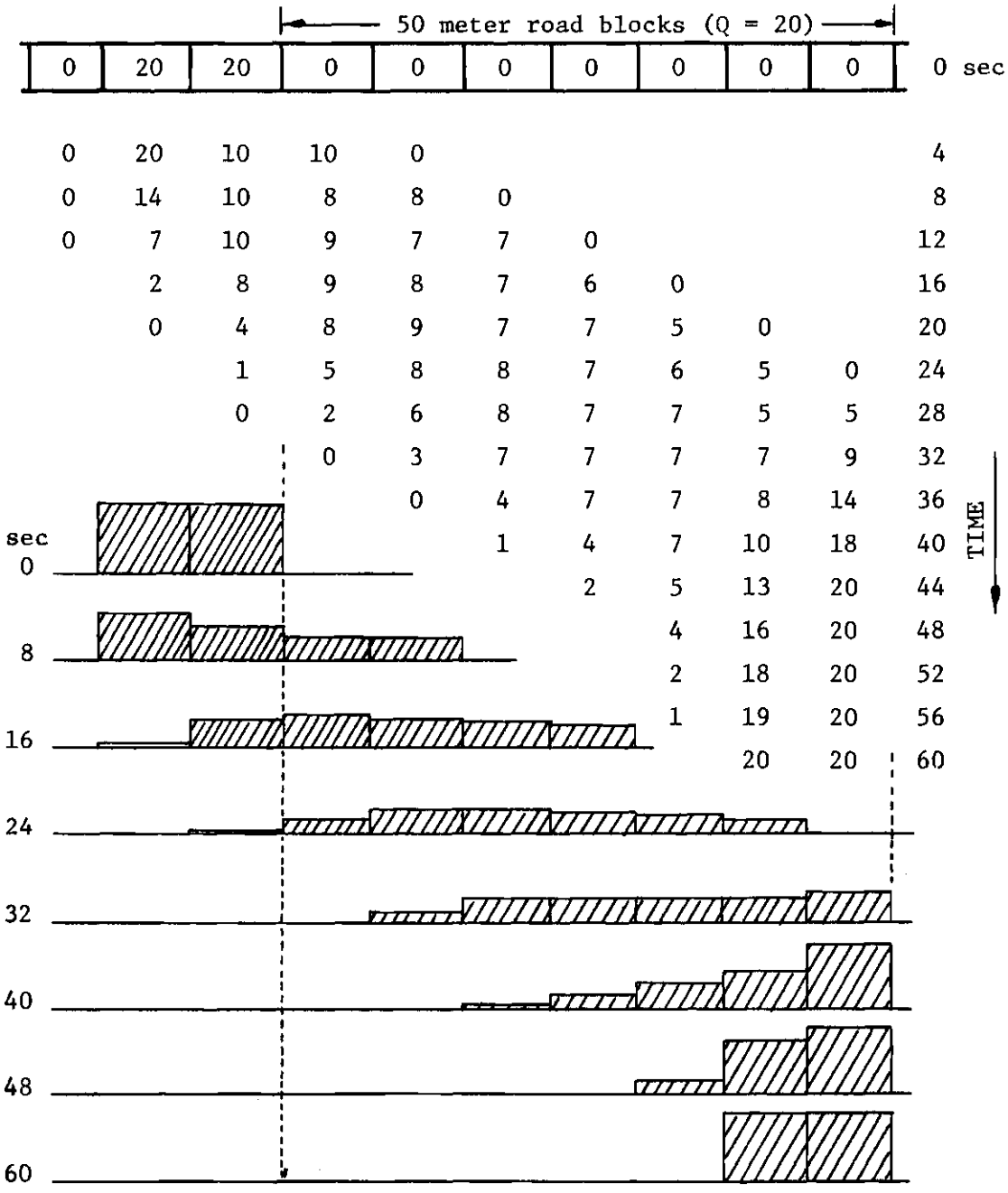


Figure 10. Platoon Dispersion Utilizing Transfer Formula

Measures of Effectiveness

The ZONE model provides output on average delay and maximum queue lengths. All statistics are printed by individual link, by intersection and by total system performance. Statistics on turning maneuvers are not recorded.

As network zones are processed, all vehicles are either advanced or delayed. If delay occurs in a zone, the number of vehicles delayed is recorded. The number of vehicles processed on a link is divided into the cumulative delay to obtain average delay. Field studies (38) indicate that this measure is adequate for light-to-medium traffic; however, average delay estimates are lower than expected in heavy traffic.

The number of vehicles in the front zone of a given link is examined whenever the traffic signal turns green. All vehicles in this zone are considered to be in the queue. If the exit zone is full, the next zone is examined. If this is full all the vehicles are considered to be in the queue and the next zone is examined. The first non-full zone marks the end of the queue. If this zone is below half full, the vehicles are considered to be moving and not in the queue; otherwise, the vehicles are the last members of the queue.

General Model Flow

The general flow of the ZONE model is illustrated in Figure 11. A complete listing of the program is given in Appendix D. The initialization phase consists of four subroutines.

EXTERN - to read data cards on external nodes.

INTERN - to read data cards on internal nodes and signals.

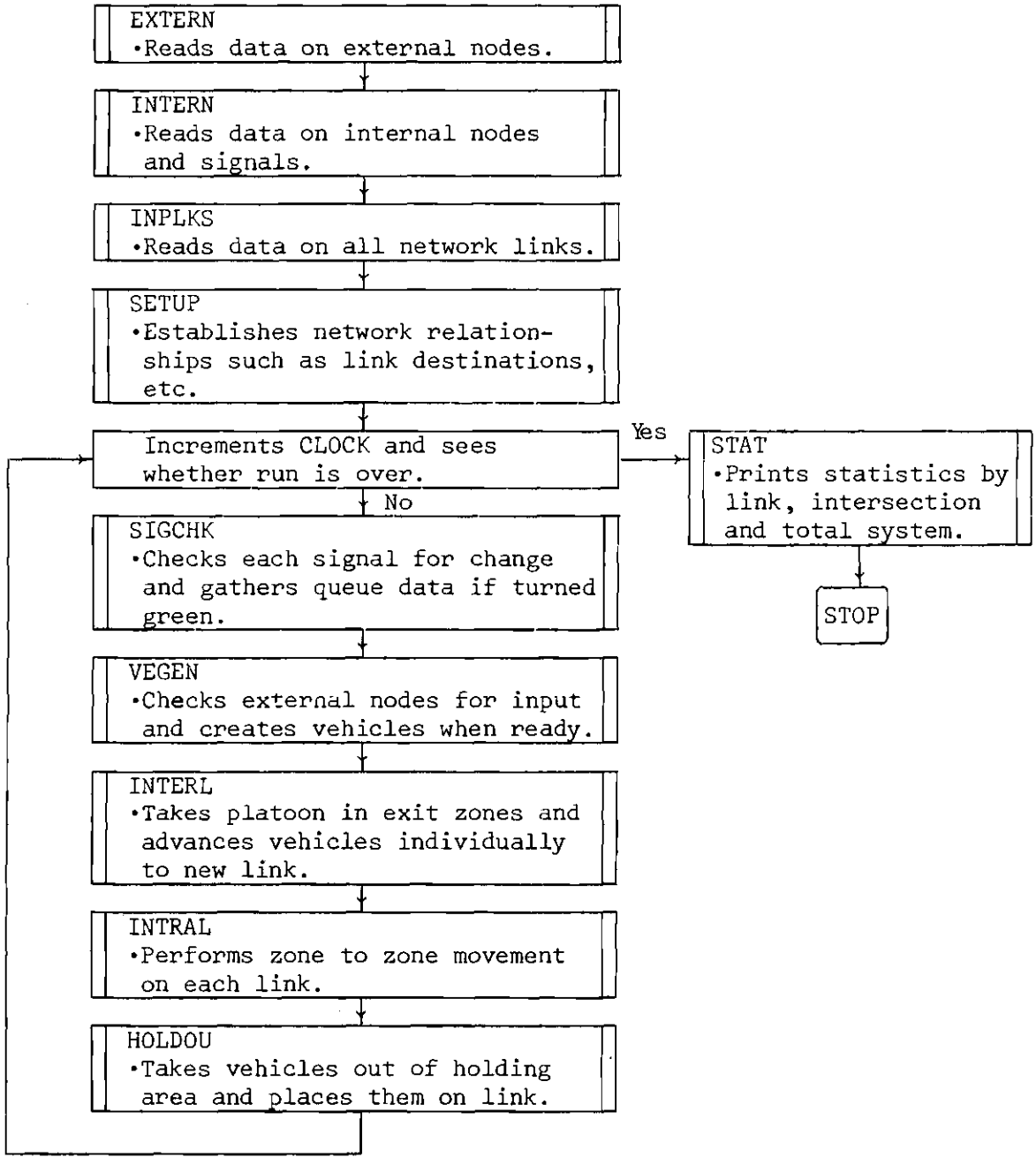


Figure 11. General Flow of ZONE Model

INPLKS - to read data cards on all network links.

SETUP - to manipulate data into desired form and schedule the first arrival for each input node.

The main body of the simulation program consists of five primary subroutines and five auxiliary subprograms.

The SIGCHK subroutine checks the signal at each intersection node to see if a change is desired. A change to green will cause the queue length to be evaluated on the appropriate input links.

The VEGEN subroutine checks each node to see if it is time to create a new vehicle. If this is desired, a new vehicle is created and the next vehicle to enter is scheduled. Because of the larger scanning times, a node may often input more than one vehicle at the same time. New vehicles are placed in the holding area of the appropriate link.

The INTERL subroutine processes vehicles in the first (exit) zone of each lane of all links while the INTRAL subroutine processes all other zones. The simplicity of these routines is such that only three supporting subprograms are necessary.

The HOLDOU subroutine removes vehicles from the link holding areas and places these vehicles on their respective links. This is done by increasing the contents of the last zone by the number of vehicles in the holding area.

The Next-Event Model

The second major departure from the more classical concepts of the continuous and unit block models is a group of models known as next-event simulation models. By concentrating on the event sequence seen by a

vehicle instead of the position sequence, researchers hoped to improve the speed of the simulation program without sacrificing the accuracy of a model in predicting the desired measure of effectiveness.

Kell (23), Thomasson (36) and Wright (43) used event-oriented models to study various types of intersections. Wohl (41) suggested this approach to study the freeway merging problem, while Francis and Lott (11) used a similar technique in studying a traffic network.

A second set of users of the next-event concept is the group of researchers who have programmed their models in simulation languages such as GPSS. Schwartz (34) developed a GPSS model to simulate a general traffic network, then applied the program to a segment of Boston. Barnes (1) developed a set of GPSS programs to simulate different types of intersections. It is from these two works that a majority of the logic has been taken to develop the last program presented in this thesis. This program will be referred to as the NEXT model.

Roadway Representation and Traffic Movement

In an event-oriented model, vehicle actions are related to specific points in time rather than precise locations on the roadway. Since each event is associated with a different position along the roadway, position does not have to be explicitly defined. At the time of an occurrence of an event, the position of the vehicle is known. Between the occurrences of two events, the vehicle is in a transition state.

Between two intersections the actual roadway representation is a sequence of three events. Figure 12 illustrates these points for a one-lane link with the associated intersections.

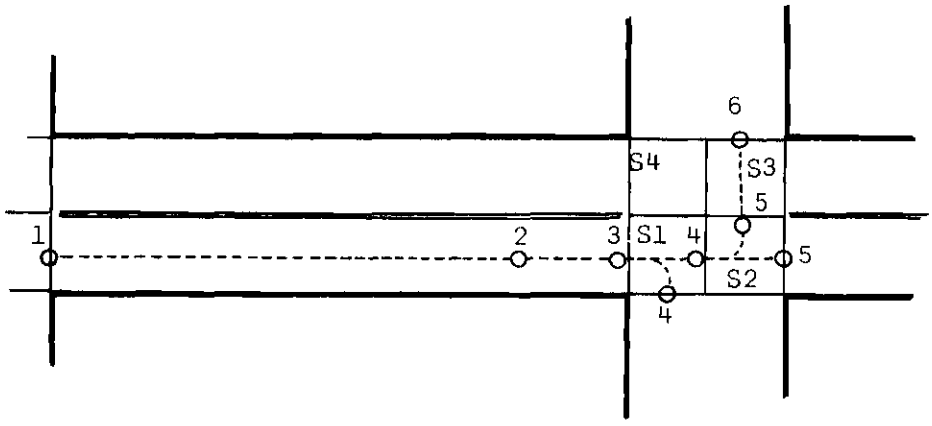


Figure 12. Event Sequence for NEXT Model

The first point on the roadway marks the entrance of the vehicle to that link. The second point of interest is the queue entrance time. If no line exists, the vehicle instantaneously represents a queue of length one. When the vehicle reaches the front of the queue, a third event occurs in that it now becomes ready to enter the intersection.

The intersection is divided into squares and the vehicle is advanced from one square to the next. Point 4 marks the two possible exit points of the S1 square. The desired turning movement will determine which exit point is encountered. Point 5 marks the exit points for the S2 square and point 6 is the exit point for S3.

Table 5 specifies the different states possible for a vehicle. Some of the states have definite time durations associated with them, while others are delay states and are system dependent. For example, the time duration for crossing an intersection square depends on the desired turning movement, the type of intersection and an acceleration factor; while travel time down a link is a function only of the desired velocity

and the link length. Time spent in a queue state is a natural result of the simulation execution.

Table 5. Possible States of Vehicle in NEXT Model

State	Event	Description	Duration
0	1-2	Free-flow in link	Formula
1	2-3	In queue, not first vehicle	Delay-system dependent
2	3	Blocked at head of queue by light	Delayed-system
3	3	Blocked by queue discharge rate or congested intersection	Delay
4a	4	Free-flow through S1	Formula
4b	4	Blocked by congestion at exit point	Delayed
5a	5	Free-flow through S2	Formula
5b	5	Blocked by congestion at exit point	Delayed
6	6	Free-flow through S3	Formula

Transition from one state to another is based on the condition of the system and the associated rules, such as maximum queue discharge rate, intersection capacity and gap availability for left-turning vehicles.

Vehicle Representation

The vehicle is represented by a set of eight computer words. These words contain:

1. Desired Speed
2. Present Link
3. Present Lane
4. Desired Turning Movement
5. Time Vehicle Entered Link

6. Current State of the Vehicle
7. Time Vehicle is Scheduled to Change States
8. Index of the Next Vehicle on the Schedule Chain.

Items 1 through 4 are common to all of the models which treat vehicles independently. At the time of link entrance, the value of the simulation clock is stored for the entering vehicle and this time is used to calculate delay when the vehicle exits the link. Item 6 indicates the state of the vehicle while item 7 specifies the scheduled time of the next state change. Item 8 will be explained in the next section.

Vehicle Movement and the Event Chain

An ordered list or "chain" of all vehicles in the network is maintained. The ordering is by scheduled event time, item 7 of the vehicle record. Vehicles with the smallest scheduled event times are at the top of the list. Delayed vehicles maintain the event times they had when they entered the delay state. Thus they are above vehicles which are scheduled to change states at some point in the future.

The next vehicle to be processed is the first vehicle on the chain with an event time greater than or equal to the current simulation clock time.

The first vehicle on the chain is identified by a pointer. This record of this vehicle indicates the next vehicle on the chain by the last data item. By going from vehicle to vehicle the chain can be constructed. The process is identical to the technique used in the CONT model to establish the ordering of the vehicles on a specific link.

Vehicles are "moved" through the network by changing their records. When the simulation clock equals the event time, the model

attempts to change the state of the vehicle. The conditions which are necessary depend on the old state of the vehicle.

Measures of Effectiveness

The NEXT model provides output on average delay, number of stops and the queue lengths. All statistics are summarized by link, by intersection and by total system performance.

Delay is defined as the difference between actual travel time and undelayed travel time. Since all state durations other than delay are calculated from deterministic functions, the undelayed travel time is easily evaluated when the desired speed, link length and intersection geometry are known. If probabilistic functions had been used, the resulting delay estimates might not be meaningful. For example, if expected travel time were used to calculate delay and a vehicle drew a fast acceleration rate, the resulting delay might have a negative value. Additionally, the field data used to estimate the necessary parameters would have to be devoid of any delay, if the simulation model is to predict this delay through system performance.

Each time the vehicle enters a delay state, a stop is recorded. The maximum number of stops a vehicle can contribute to the cumulative total stops for a given link is a function of the turning movement desired. A left-turning vehicle would stop as many as three times while a right-turning vehicle could encounter one stop at most.

The NEXT model is the only model which continuously maintains statistics on the status of a queue. Since queue entrance time is not scheduled until the full length of the link has been traversed, this

statistic should be somewhat lower than that measured by the other models. A more complex algorithm might be used to calculate travel time to the end of the queue if this statistic were used as a primary measure of effectiveness. Such an algorithm would consider queue length, vehicle length, discharge rate and current state of light and time of next light change.

General Model Flow

The general flow of the NEXT model is illustrated in Figure 13. A complete listing of the program is given in Appendix E. The initialization phase consists of four subroutines:

EXTERN - to read data cards on external nodes.

INTERN - to read data cards on internal nodes and signals.

INPLKS - to read data cards on all network links.

SETUP - to manipulate data into desired form and schedule the first arrival for each input node.

The main body of the simulation program consists of three primary subroutines and nine auxiliary subprograms. As indicated in Figure 13, the organization and processing of these subroutines differs drastically from that of the other three models.

The main program examines three subroutine clocks to see when the next state change of some system component is going to occur. These clocks are:

SIGTIM - the time of the next signal change.

VEGTIM - the creation time of the next new vehicle.

MOVTIM - the next vehicle event time.

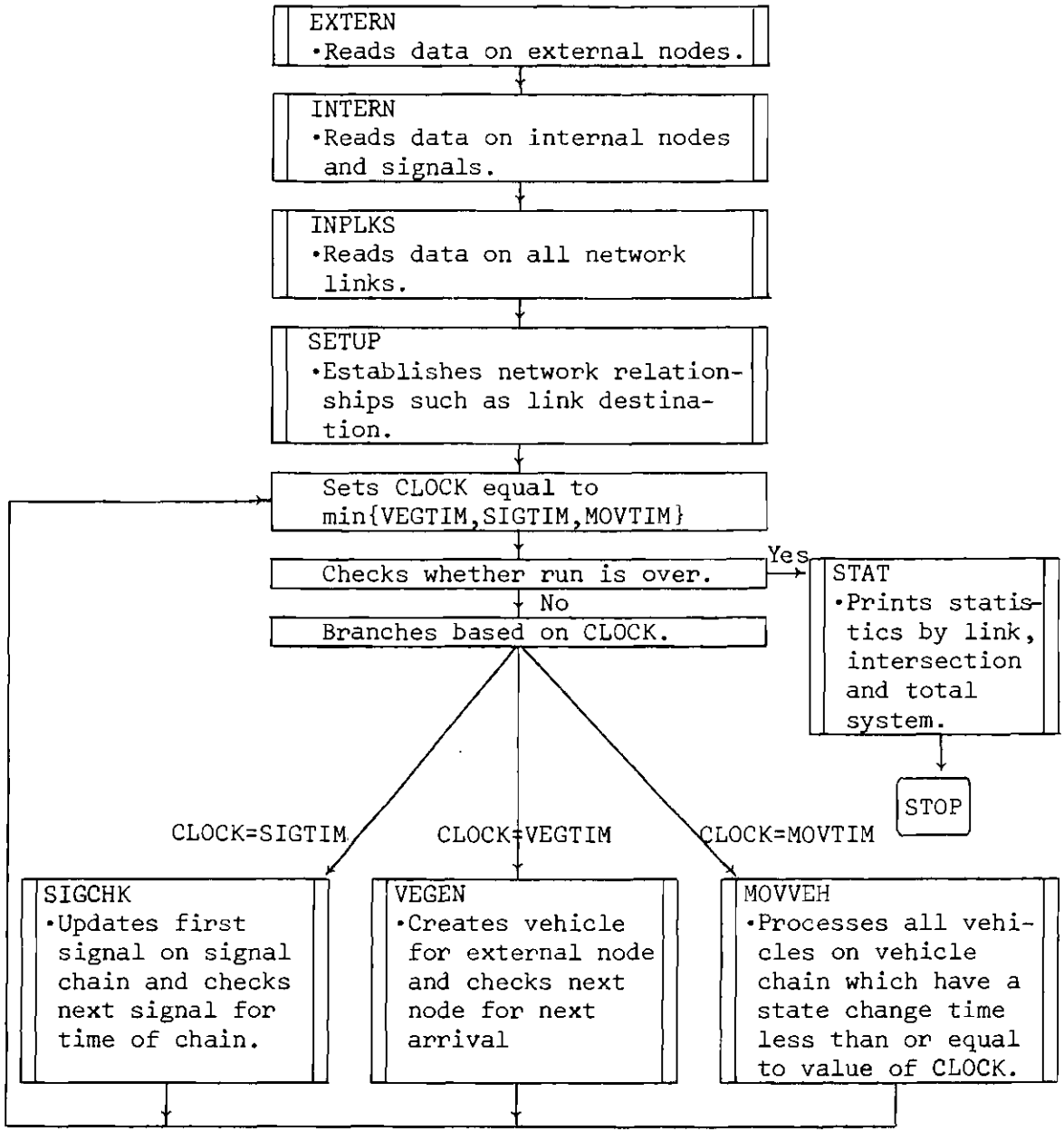


Figure 13. General Flow of NEXT Model

The simulation clock is advanced to the minimum of these three times. Control is then passed to the appropriate subroutine.

The SIGCHK subroutine changes the state of traffic signals when it is desired. Like the vehicles, signals are maintained on a chain in an event time ordering. This feature eliminates the necessity of checking each signal to see if a change is necessary.

The VEGEN subroutine creates vehicles and places them on the desired link. Because of the method of processing vehicles, no holding area is required in this model.

The MOVVEH subroutine manipulates vehicles which are scheduled to be processed at the current clock time or are in a delay state. If a congested link has prohibited a newly created vehicle from entering the system, the VEGEN routine is called to see if the congestion has dissipated.

CHAPTER IV

COMPARISON AND EVALUATION OF MODEL PERFORMANCE

Once the four models were programmed and debugged, an extensive testing program was undertaken to compare the behavior of the models under different situations.

Model Verification and Validation

Before the actual comparison of the models was considered, a number of computer runs were devoted to establishing the credibility of the programs. This operation was divided into two phases:

1. The verification stage, which was concerned with comparing the operations of the program against the desired logic of the model.
2. The validation stage, which was concerned with comparing the responses of the models with expected behavior of a physical network.

Model Verification

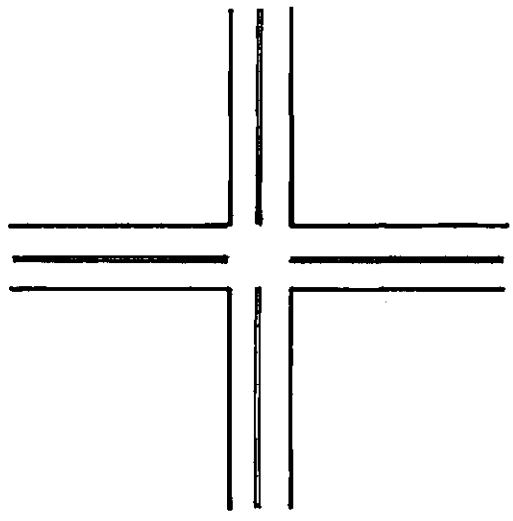
A series of tests were performed to analyze the logical behavior of each of the models under different traffic conditions. Additional tests were performed on specific components that were common to all of the models. These tests were not intended as a means of comparing the models; rather, they were performed to show that each of the models produced acceptable vehicle behavior.

Realism of Traffic Movement. A number of output statements were added to each of the programs. These statements provided detailed

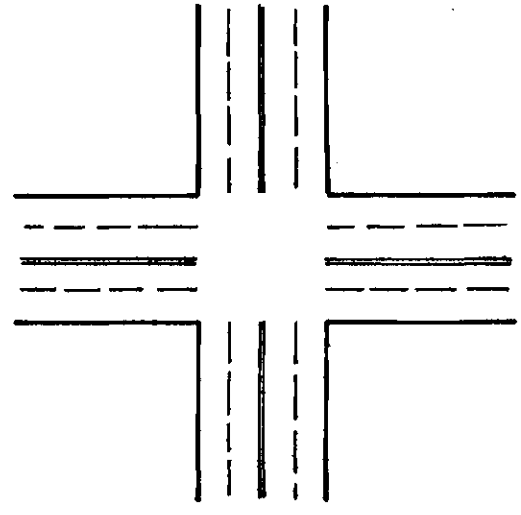
information on every movement or decision of all vehicles in the system. Through an analysis of this output, the complete movement of each vehicle could be traced throughout its sojourn in the system. Three test networks were constructed as illustrated by Figure 14. Each model was run for 120 seconds of simulation time and the history of the vehicle movements was obtained. Cork boards, with drawings for each network, and map pins were used to visually trace each vehicle in the network. The use of this information resulted in several refinements to the programs.

Although these verification tests alone are not sufficient evidence to establish the credibility of the models, they do provide confidence in the basic sequence of actions followed by vehicles under different situations. Inoperative forms of the output statements used in this phase of testing have been retained in the program listings. A letter C and four asterisks precede the actual WRITE commands.

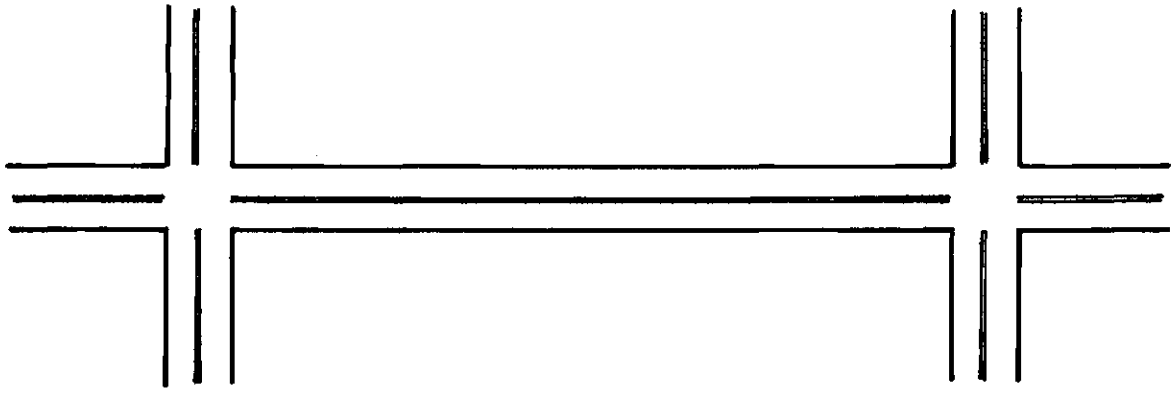
Pseudo-Random Number Tests. Although the method used by all models for generating pseudo-random numbers is considered acceptable, a series of tests were conducted to insure that proper parameters would be used in the actual runs. Utilizing different random number seeds the following procedure was employed. First 5000 pseudo-random numbers were generated. Next the numbers were grouped into classes with an interval width of 0.1 and a chi-square goodness-of-fit test performed. Then the sequence was subjected to a runs-up and runs-down test. Finally, serial correlation analysis for lags of 2, 4, 6 and 10 were used to test randomness.



a. Two-Lane by Two-Lane Intersection



b. Four-Lane by Four-Lane Intersection



c. Network of Two Two-Lane by Two-Lane Intersections

Figure 14. Model Verification Networks

All tests were performed at a 5 per cent level of confidence. Only those seeds which passed all tests were retained for use in the actual experimental runs.

Vehicle Generation Testing. All models used the same method of generating exponentially distributed interarrival times. To test this generator, each seed selected by the previous tests was used to schedule arrivals for rates of 200, 300, 400, 500 and 600 vehicles per hour. For each volume, a five-hour sample was taken and the resulting interarrival time distribution was tested with a chi-square goodness-of-fit test. Additionally, five-minute traffic counts were obtained and tested against the appropriate Poisson distribution. All seeds passed each test at a 5 per cent level of significance.

Model Validation

After the logic of each program had been checked as described above, the comparison of model performance with expected behavior was undertaken. It is realized that a complete assessment of the accuracy of the simulation programs can be accomplished only through the application of an extensive field testing program. However, this type of validation is a most difficult undertaking even when unlimited resources are available. It was hoped that the models could be compared to some of the delay studies reported in the literature.

Berry (3) has published the results of extensive studies in the field measurement of delay. However, the published report on these studies is not complete in the description of the roadway sections and vehicle characteristics. Kell (24) encountered difficulty with the time

and manpower required to obtain field data, and with a more important limitation. The simulation technique which he used, like the ones used here, evaluates total system delay. Field studies are generally limited to measurement of stopped delay, ignoring delay accumulated in the process of accelerating and decelerating. The basic problem in validation is simply that information of the type readily obtained from the model is extremely difficult to measure in the field.

Webster's Equation. Webster (40) has developed a relationship between delay and saturation using a combination of theoretical and empirical techniques. The general delay equation is

$$d = \frac{c(1-\lambda)^2}{2(1-\lambda x)} + \frac{x^2}{2q(1-x)} - 0.65\left(\frac{c}{q^2}\right)^{1/3} x^{(2+5\lambda)}$$

where d = average delay per vehicle on a particular approach arm of the intersection.

c = cycle length in seconds.

λ = portion of the cycle which is effectively green.

q = flow rate in vehicles per second.

x = the degree of saturation. This is the ratio of the actual flow to the maximal flow which can pass through this approach arm of the intersection.

The first term represents delay for traffic arriving at a uniform rate, while the second term represents added delay caused by the random nature of arrivals. The third term represents an empirical correction term added to improve the agreement of the equation with actual road conditions.

To test the validity of the simulation models, the delay estimates of all four models were compared to that computed by the Webster equation for various combinations of flow rates and turning maneuvers. The flow rates covered 200, 300, 400, 500 and 600 vehicles per hour. For each flow rate, the turn percentages indicated in Table 6 were specified.

Table 6. Turn Percentages in Validation Tests

Flow Set	Right	Left
I	0%	0%
II	5	5
III	10	10

All terms in the Webster equation are easily obtained with the exception of maximal flow. The maximal flow was determined in this study in accordance with the 1965 Highway Capacity Manual (20). The intersection approach considered was a two-lane, two-way street with no parking. Maximal flow was determined by assuming a load factor of unity and applying the appropriate turning maneuver correction factors.

Since the Highway Capacity Manual does not take into account opposing flow for left turns, an adjustment factor proposed by Collins (10) was used. In runs where left-turning traffic was present, the flow obtained from the Highway Capacity Manual was multiplied by $e^{-qt}(qt+1)$ where q is the opposing flow in vehicles per second and t is the average

acceptable gap in seconds. This adjusted maximal flow was then used to calculate average delay by the Webster equation.

Results of Tests. A typical two-lane, right-angled intersection was chosen for validation testing. The cycle length was set at 60 seconds with 32 seconds of green time on the main street. Main street flow rates covered 200, 300, 400, 500 and 600 vehicles per hour in each direction. The side street flow rate was held constant at five vehicles per hour. Three sets of flow rates were used in order to test the sensitivity of the models to the turn percentages indicated earlier.

Four runs of each simulation model were performed. Each run consisted of 30 minutes of simulation data time preceded by a 3-minute warm-up time. For each run the average delay per vehicle on each main approach link was computed and the two data points recorded. For each combination of model, flow rate and turning maneuver, the eight data points were averaged and a confidence interval computed. Table 7 gives the results of the simulation model comparisons to the delay time computed by the Webster equation. The adjusted equation, suggested by Collins, is also included for reference. However, all statistical comparisons were performed with the more accepted Webster equation. Figures 15, 16 and 17 present a graphical representation of these data.

The overall agreement between the model results and the Webster equation are satisfactory. The CONT model and UB model produced only one rejection each out of 15 independent applications of a confidence interval test. The results of these two models also compare favorably in the low and medium flow ranges. A wider dispersion is noted in the

500 to 600 vehicles per hour range. The ZONE model is significantly different from the Webster equation at four points. A general trend at underestimating delay in all but the high flow rates can be observed from the graphs of the data. The NEXT model was also found to be significantly different from the Webster equation at four points. In general this model tended to overestimate delay. A more favorable agreement was found between the NEXT model and the adjusted equation proposed by Collins.

Table 7. Average Delay per Vehicle for Validation Study

		MODEL PREDICTIONS				EQUATIONS	
	FLOW	CONT	UB	ZONE	NEXT	Webster	Adjusted
No Turns	200	9.54	9.26	8.41	9.71	9.0	9.0
	300	10.60	10.37	9.31	11.23	9.9	9.9
	400	10.95	10.95	9.52	11.76	11.0	11.0
	500	11.84	11.88	10.03*	12.83	12.2	12.2
	600	12.81	17.19	11.73*	15.80	13.9	13.9
5% Right 5% Left	200	9.36	8.81	8.70	9.99	9.6	9.6
	300	11.13	11.03	10.05	12.47*	10.9	11.2
	400	12.20	13.39	11.41	15.42*	12.6	13.5
	500	15.53	18.55	17.09	19.89*	15.0	18.4
	600	21.77	53.36*	35.07*	34.07	20.6	77.8
10% Right 10% Left	200	9.58	9.44	9.01*	10.95	10.6	10.7
	300	11.69	11.73	10.75*	14.57	12.8	13.3
	400	14.08	15.73	14.08	22.78*	16.4	19.0
	500	21.21*	33.53	23.36	26.24	28.5	-
	600	51.69	86.62	66.28	66.38	-	-

*Significant at a 5 per cent level of confidence.

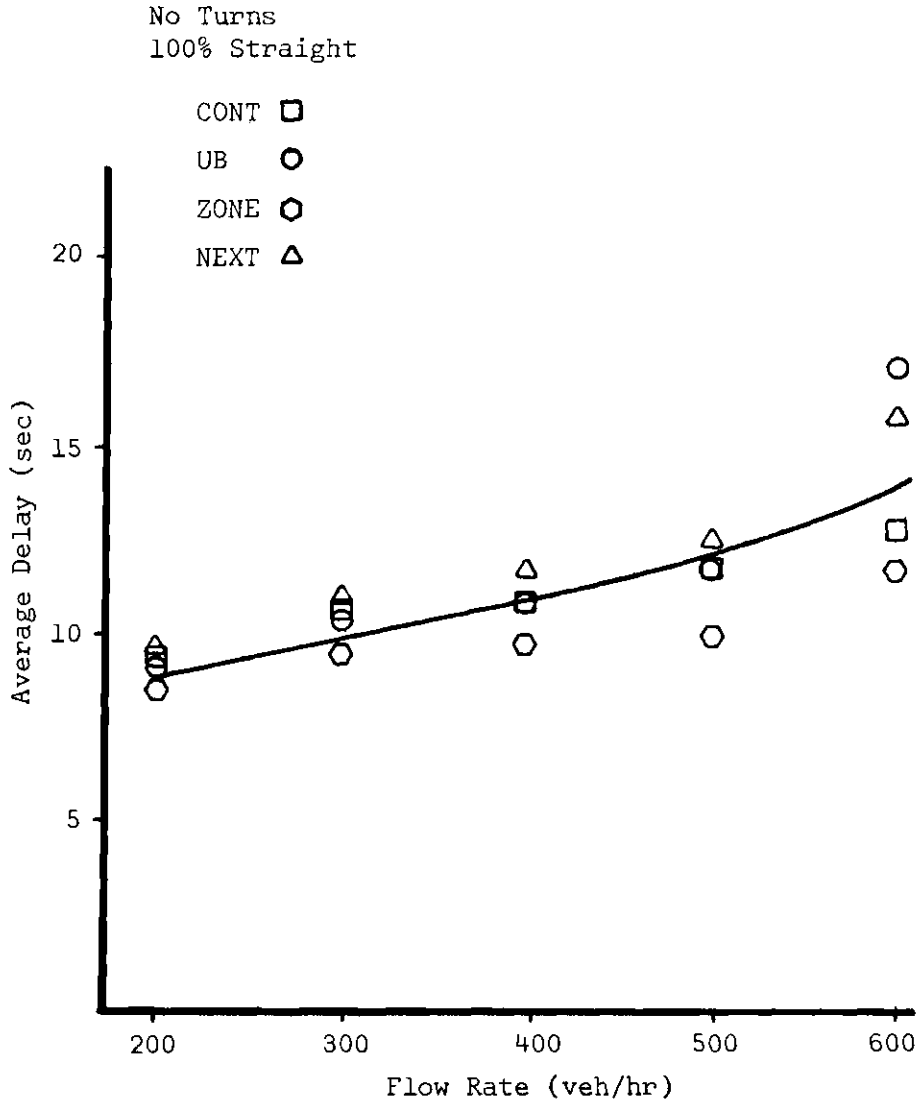


Figure 15. Comparison of Webster Equation to Predictions from Models for No Turns

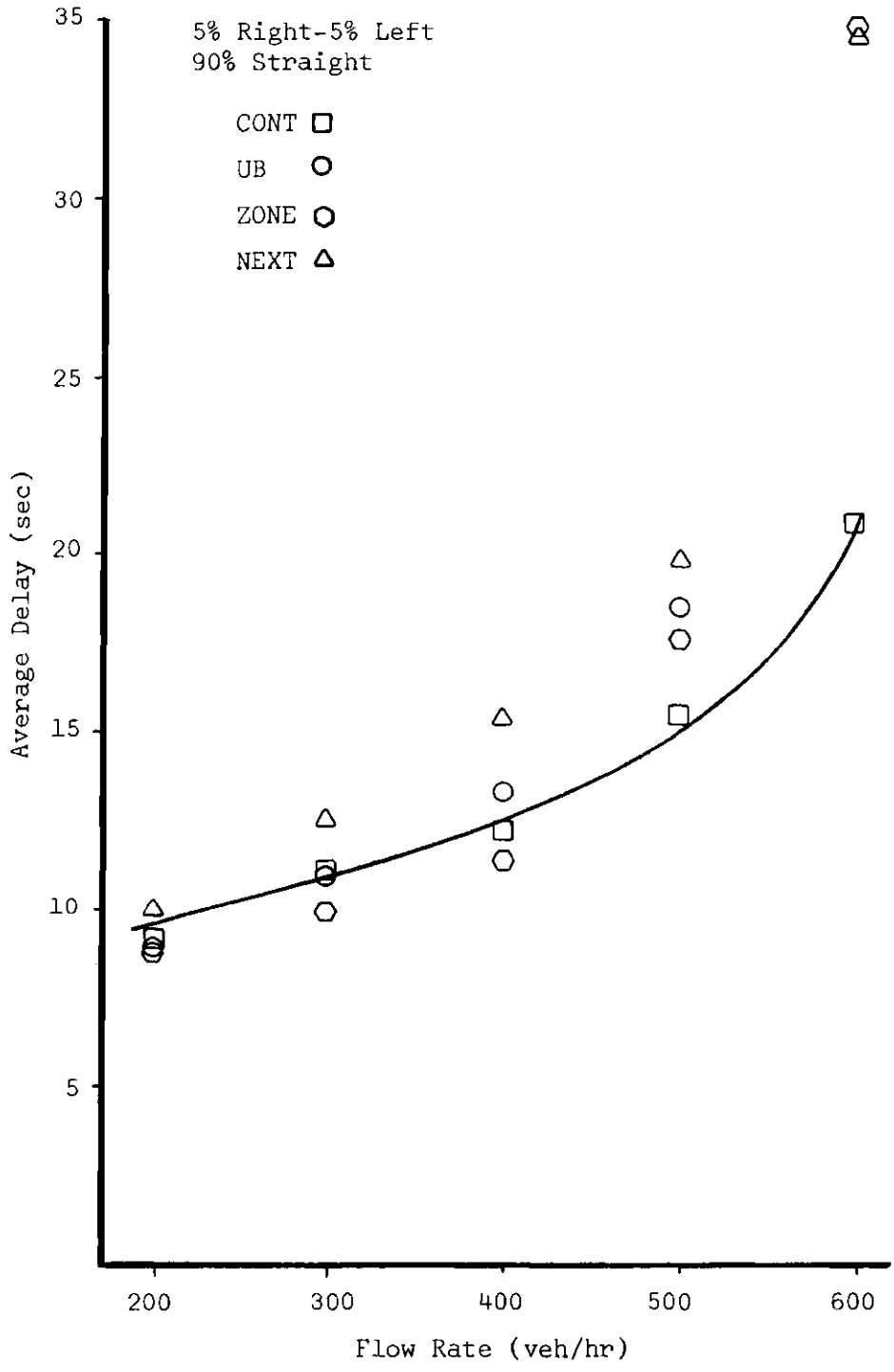


Figure 16. Comparison of Webster Equation to Predictions from Models for 5% Left-5% Right Turns

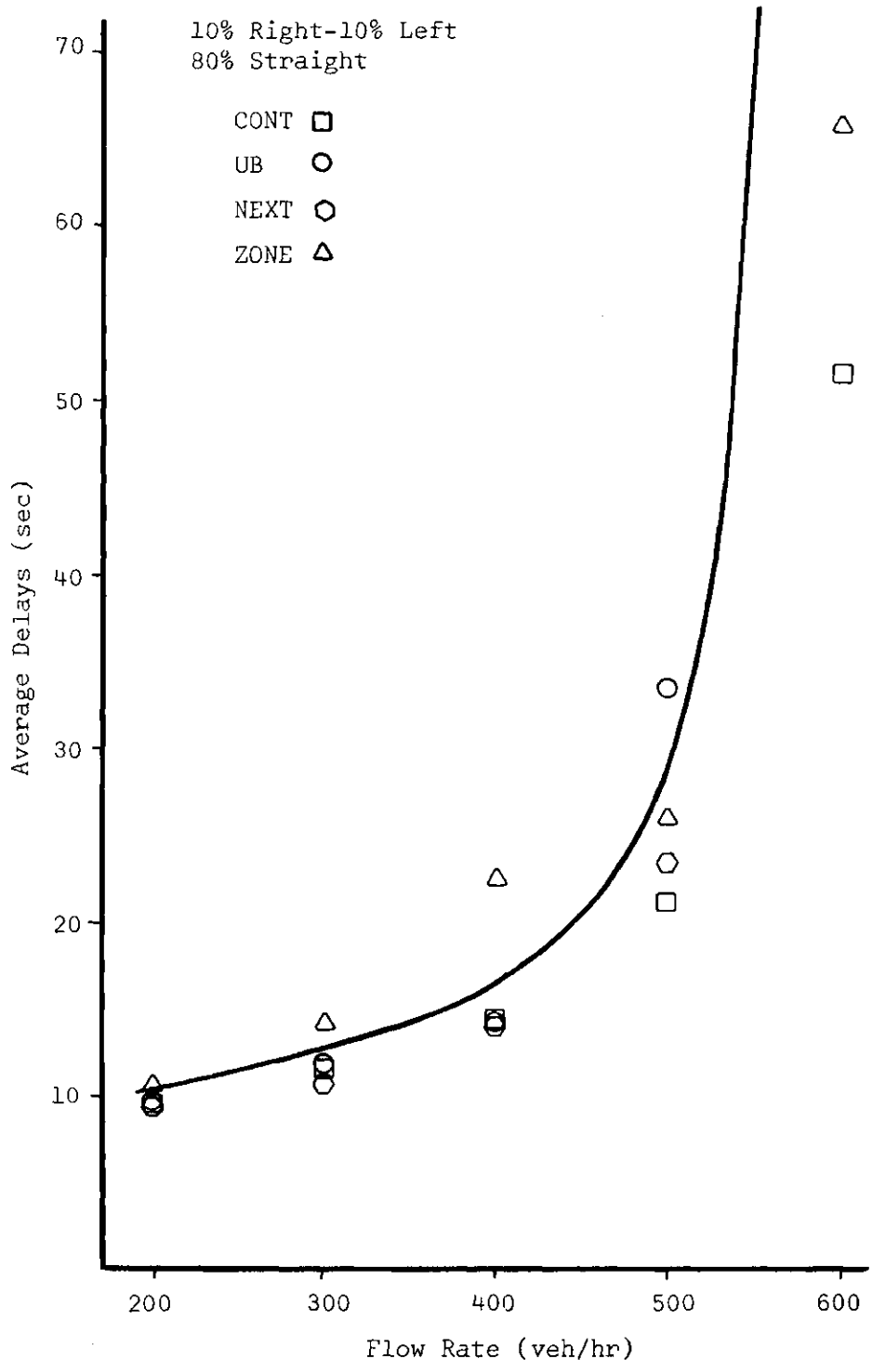


Figure 17. Comparison of Webster Equation to Predictions from Models for 10% Left-10% Right Turns

Network Performance Comparison

After confidence had been established in the performance of the individual models, a series of computer runs were undertaken whereby the performance of the models could be further analyzed and compared. Various service volumes as well as network configurations were considered in order to ascertain their influence on model behavior.

Description of Test Networks

To test the effects of network configuration on model performance, it was decided to run each model on three different street patterns--a single intersection, an arterial pattern and a closed network.

Network Considered. To approximate an urban situation, each intersection was assumed to be controlled by a fixed time traffic signal and all link lengths were less than 800 feet. A link-node diagram of each network configuration is given in Figure 18. The numbers in parentheses are the link lengths. All streets are four lanes wide with no curbside parking permitted.

To facilitate comparing the performance of the models on different configurations, the three networks are nested; the single intersection is contained in the arterial pattern which is part of the closed network. Since the input links to the intersection at Third Street and Main Street (see Figure 18) have the same characteristics in all configurations, the performance measures for these links formed the basis for model comparison. In order to compare delay predictions from the three configuration, it was essential that some common street segment be included in each configuration. The intersection of Main Street and

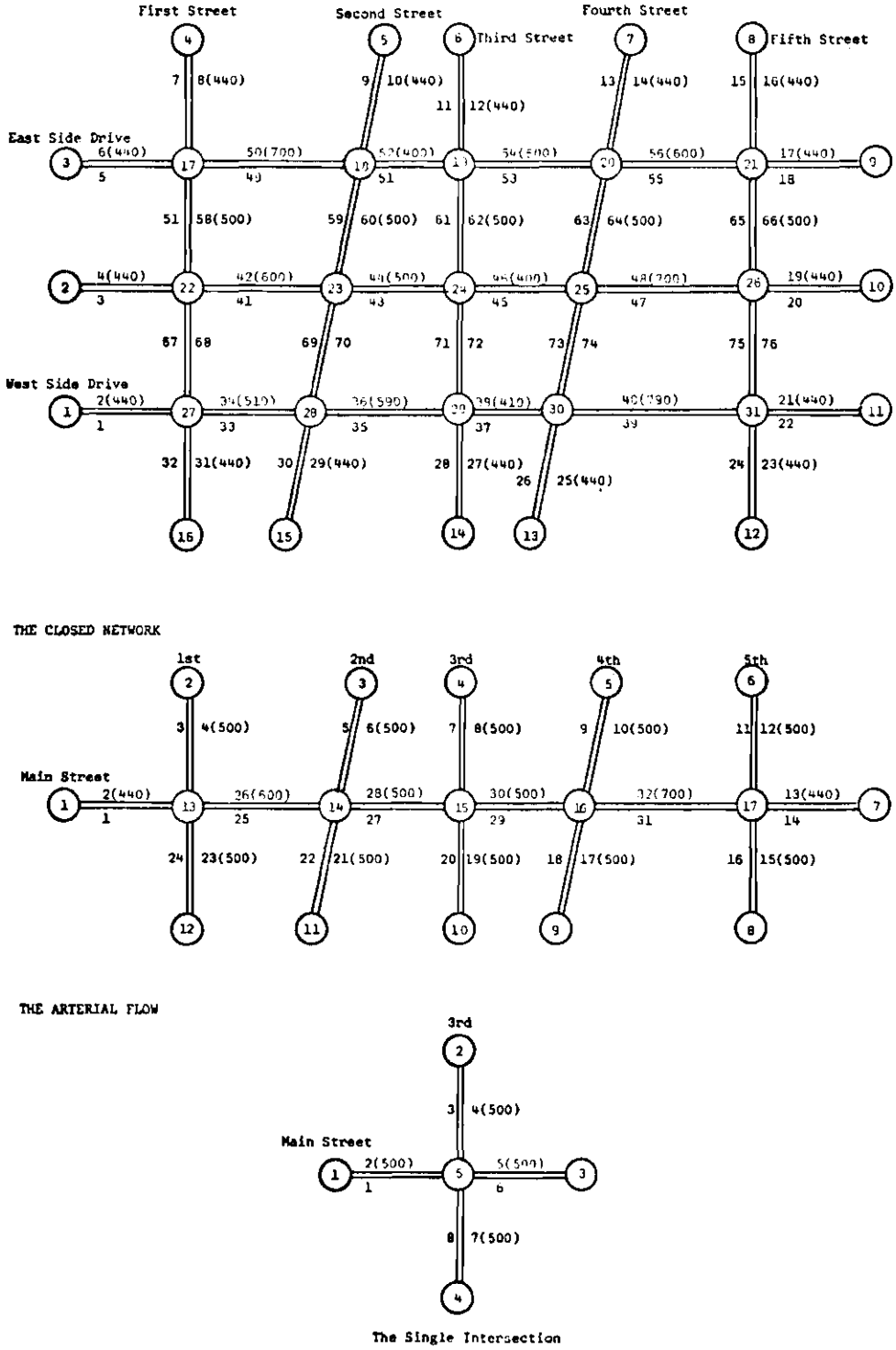


Figure 18. Link-Node Diagrams of Test Networks

Third Street forms such a base.

Selection of Signal Settings. In order to insure that the signal settings were not a major source of delay, a progressive timing pattern was determined for each of the arterial flows. Without progression, the capacity of an individual intersection may be decreased because vehicles may be unavoidably detained due to poor sequencing of the signals. With proper progression, the main platoon will witness delays only when starting at the first signal. If the progression speed is maintained, these vehicles can then pass through the other signals without further delay.

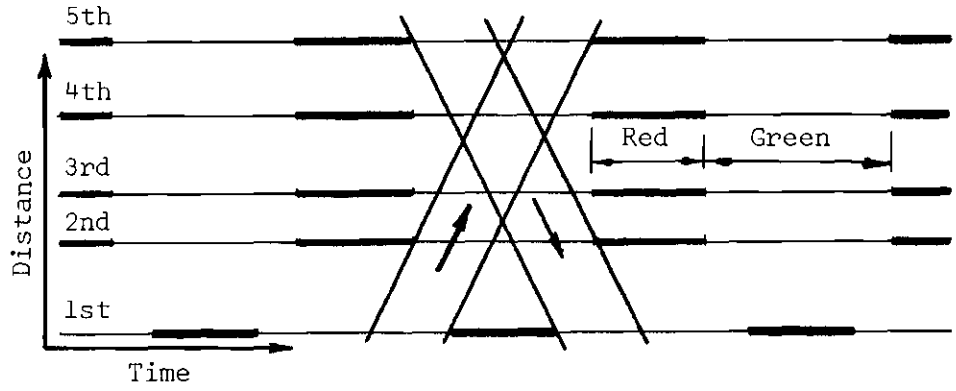
To determine the progressive timing patterns, the algorithm proposed by Morgan and Little (30) was programmed. This algorithm determines the maximal equal bandwidths* for opposing flows on a two-way arterial. The program was run for each arterial street for signal cycle times ranging between 60 and 90 seconds. Since the 90-second cycle produced the largest bands on each artery, it was selected as the network cycle length. Figure 19 illustrates the resulting time-space diagram for the three main streets in the connected network.

Volume Considerations. For the purposes of the simulation input, traffic volumes are specified at the external nodes only. Internal volumes are obtained as a result of the simulation process.

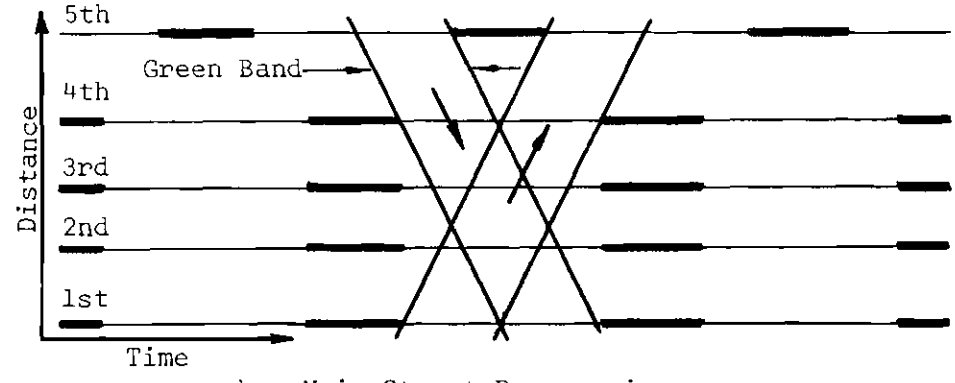
A procedure was established to insure equivalent volumes, for all network configurations, on the four links leading to the main

---

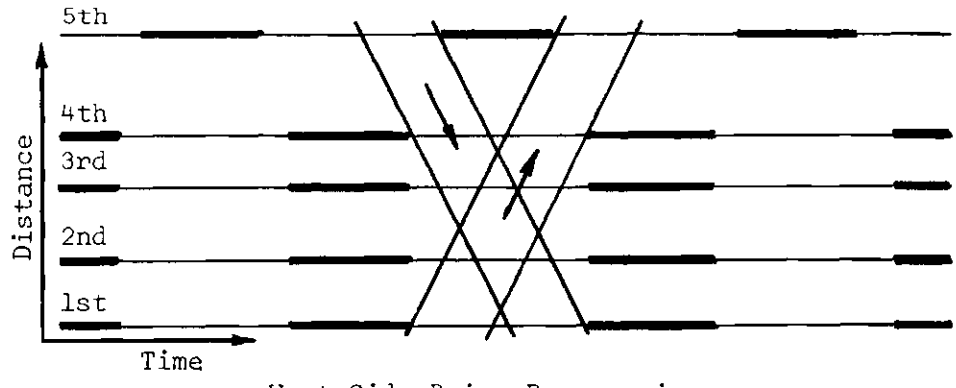
*The bandwidth is defined as the size (in seconds) of the largest platoon which can pass through a series of signals without stopping.



a. East Side Drive Progression



b. Main Street Progression



c. West Side Drive Progression

Figure 19. Time-Space Diagrams for 90-Second Cycle

intersection. Let

$$p_{11}x_1 + p_{12}x_2 + \ldots + p_{1n}x_n = V_1$$

$$p_{21}x_1 + p_{22}x_2 + \ldots + p_{2n}x_n = V_2$$

$$\vdots \qquad\qquad \vdots$$

$$p_{n1}x_1 + p_{n2}x_2 + \ldots + p_{nn}x_n = V_n$$

where x_i is the volume per hour on the *ith* link,

p_{ii} $\begin{cases} \text{is +1 if the link is a source link, or} \\ \text{is -1 if the link is an internal link or a sink link,} \end{cases}$

p_{ij} is the probability of going to the *ith* link from the *jth* link $i \neq j$,

V_i is the external source volume for the *ith* link, and

n is the number of links.

For any given set of external volumes $\{V_i\}$ the internal volumes for each link $\{x_i\}$ can be calculated by solving the simultaneous equations.

To arrive at three different flow levels for the various network configurations, a number of external source volume combinations were considered for the closed network. The simultaneous equations for each volume combination were solved and three combinations were selected which yielded light, medium and heavy service volumes for the principal intersection. Next the source links for the arterial pattern and the single intersection were considered. The theoretical volumes $\{x_i\}$ determined in the closed network were used as source volumes in the

smaller networks. Thus, the four links of importance theoretically are being subjected to the same volumes of traffic for each network configuration. Appendix F lists the volumes projected for the closed network. It should be noted that the heavy traffic has been chosen at a level below 500 vehicles per hour per lane in order to insure stability in the network flow.

Test Results

Two computer runs were made for each combination of simulation model, network configuration and flow rate. Each run consisted of 45 minutes of simulated data gathering time. The warm-up times were dependent on network configuration as given in Table 8.

Table 8. Model Warm-up Times

Network Configuration	Warm-up Time (min)
Intersection	5
Arterial	10
Network	15

Model Performance. During each run, the average delay per vehicle for the main intersection was computed and recorded. The results of the test runs are given in Table 9. A three-factor analysis of variance of these data is presently in Table 10. All effects are considered as fixed effects and the assumption of an analysis of variance test appear to apply to this experiment.

Table 9. Average Delay per Vehicle at Main Intersection of Test Networks

Configuration	Flow	MODEL			
		CONT	UB	ZONE	NEXT
Intersection	Light	16.73 16.53	16.08 15.13	15.01 14.90	16.60 15.63
	Medium	21.81 20.96	19.43 18.94	17.65 19.19	18.62 18.54
	Heavy	31.39 46.10	28.05 46.46	32.25 37.80	25.87 23.54
Arterial	Light	15.30 14.87	11.74 12.01	10.98 11.21	11.39 10.86
	Medium	18.62 19.05	14.82 13.98	13.80 13.12	13.06 13.18
	Heavy	31.65 29.74	37.76 31.81	23.18 27.38	17.26 21.19
Network	Light	12.27 12.13	7.33 6.84	6.65 6.32	6.70 6.28
	Medium	17.67 14.70	9.76 8.50	8.89 9.47	7.73 7.83
	Heavy	35.01 25.03	27.30 23.50	17.22 24.30	15.36 14.09

Table 10. Analysis of Variance of the Model Comparison Data

Source of Variation	Sum of Squares	df	Mean Square
Models (M_i)	557.49	3	185.83**
Networks (N_j)	1030.27	2	515.13**
Flows (F_k)	3481.75	2	1740.87**
M × N	54.70	6	9.12
N × F	331.52	6	55.25**
N × F	28.79	4	7.20
M × N × F	37.68	12	3.14
Error	423.98	36	11.78
TOTAL	5946.19	71	
**Significant at 1%.			

It can be seen that all main effects are highly significant. With respect to the first factor, this implies that the type of model used does influence the delay prediction. The data, presented in Table 9, indicate that the ZONE and NEXT models predicted delays below that of the CONT and UB models for all combinations of flow and configuration except the light flow on the single intersection. The validation test somewhat indicated this possibility for the ZONE model; however, the NEXT model predicted higher delays than expected in single-lane tests.

The assumptions regarding vehicle interactions and accelerations are thought to be a primary reason for these low estimates of delay. In the ZONE model vehicles are either moving at "free speed" or standing still. Such an assumption requires instantaneous acceleration to free speed. Furthermore, there is no interaction between vehicles advancing

in the same direction. The NEXT model does not take into account any vehicle interaction on the link during the vehicle's journey to the queue state. Furthermore, acceleration delay considerations, due to a low initial speed, are minimal when the vehicle is scheduled to travel down the link. This latter problem would not affect delay calculations in modeling a single intersection, since only incoming links to the intersection are significant and initial speeds of new vehicles are set equal to their desired speeds.

Predictions of delay from the UB model usually fall between those from the CONT model and the ZONE and NEXT models. The CONT model predicted higher delays than the other models in all but one combination of configuration and flow.

The effect of network configuration on vehicle delay was significant. This result is reasonable when the advantages of the progressive signal system are considered. Vehicles, which enter the "green band" at an exterior intersection, could pass through the system without delay. Other vehicles, which should have stopped at the main light, were delayed at a previous intersection and forced to wait for the band. As configuration was changed from an arterial roadway to the closed network, the side street progression further reduced delay.

As expected, flow rates had a highly significant effect on delay. As flow increased, left-turning vehicles found smaller gaps and were forced to wait, queues were longer and some vehicles were forced to wait through more than a partial cycle of the light. Since average link speed is reduced, vehicles can no longer stay in the green band and hence some

of the advantages of the progressive signal system are lost.

A highly significant interaction was found between models and flows. This result again agrees with the graphical results obtained in the validation testing. Certain models were more sensitive to changes in flow rate than other models. No other interactions were statistically significant.

In addition to the above analysis, each model was subjected to a single long simulation run in order to obtain a time series of the average delay occurring on specific links of the network. The intersection configuration under medium flow was selected for this test. A five-hour sample run was made. The North and South approaches to the main intersection were chosen as the links of interest. Every 90 seconds, the delay per vehicle exiting the link was computed and recorded. The average delay encountered in each time interval was computed and the results were punched on computer cards for further analysis.

The sequence of observations $\{d_j\}$ for a link constitutes a time series of 200 observations. The first ten observations were discarded to allow for system loading. An analysis of the autocorrelation of the series was then conducted to examine the dependence of delay at time t with some previous delay. If a higher than average observed delay tends to be followed by another higher than average delay observed k time periods later, the autocorrelation between d_t and d_{t+k} is positive. Similarly, if a lower than average delay is followed by another lower than average delay at a lag of k time periods, the autocorrelation between d_t and d_{t+k} is negative. An estimate of the autocorrelation for

a lag of k time units can be obtained from the sample autocorrelation function

$$\hat{\rho}_k = \frac{\frac{1}{N-k}\sum_{t=1}^{N-k}(d_t-\bar{d})(d_{t+k}-\bar{d})}{\frac{1}{N}\sum_{t=1}^{N}(d_t-\bar{d})^2} \qquad k = 0,1,\ldots,K$$

where N is the total number of observations.

k is the lag size.

d_t is the 90-second average delay observed at period t.

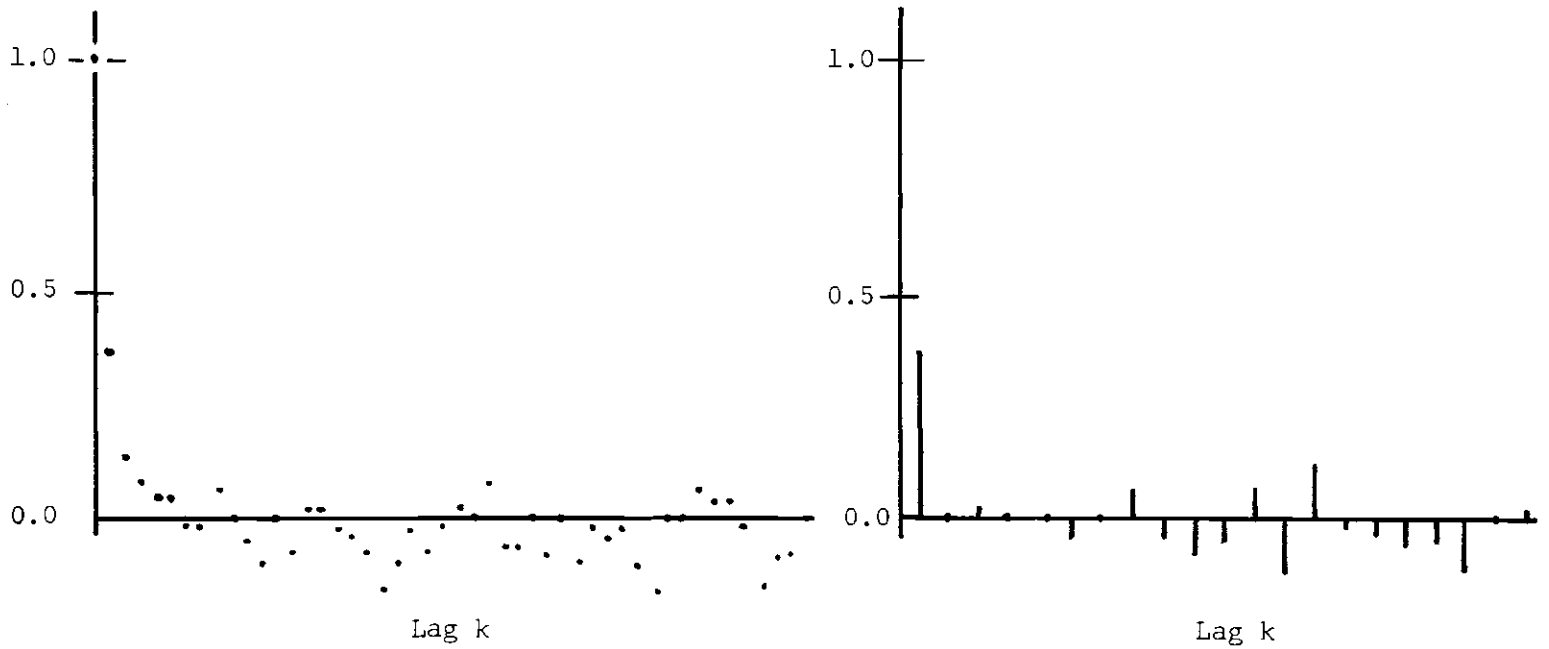
K is some arbitrary upper limit on lag, usually $\leq \frac{N}{4}$.

$\bar{d}$ is the average of the $\bar{d}_t$'s.

The partial autocorrelation or conditional correlation function is also estimated using the recursive method outlined in Box and Jenkins (6). This function is also helpful in identifying the type of dependencies which exist.

Typical plots of a sample autocorrelation function and the associated partial autocorrelation function are presented in Figures 20 and 21 for the North approach link and South approach link, respectively. The sample autocorrelation functions for all four models are given in Appendix G.

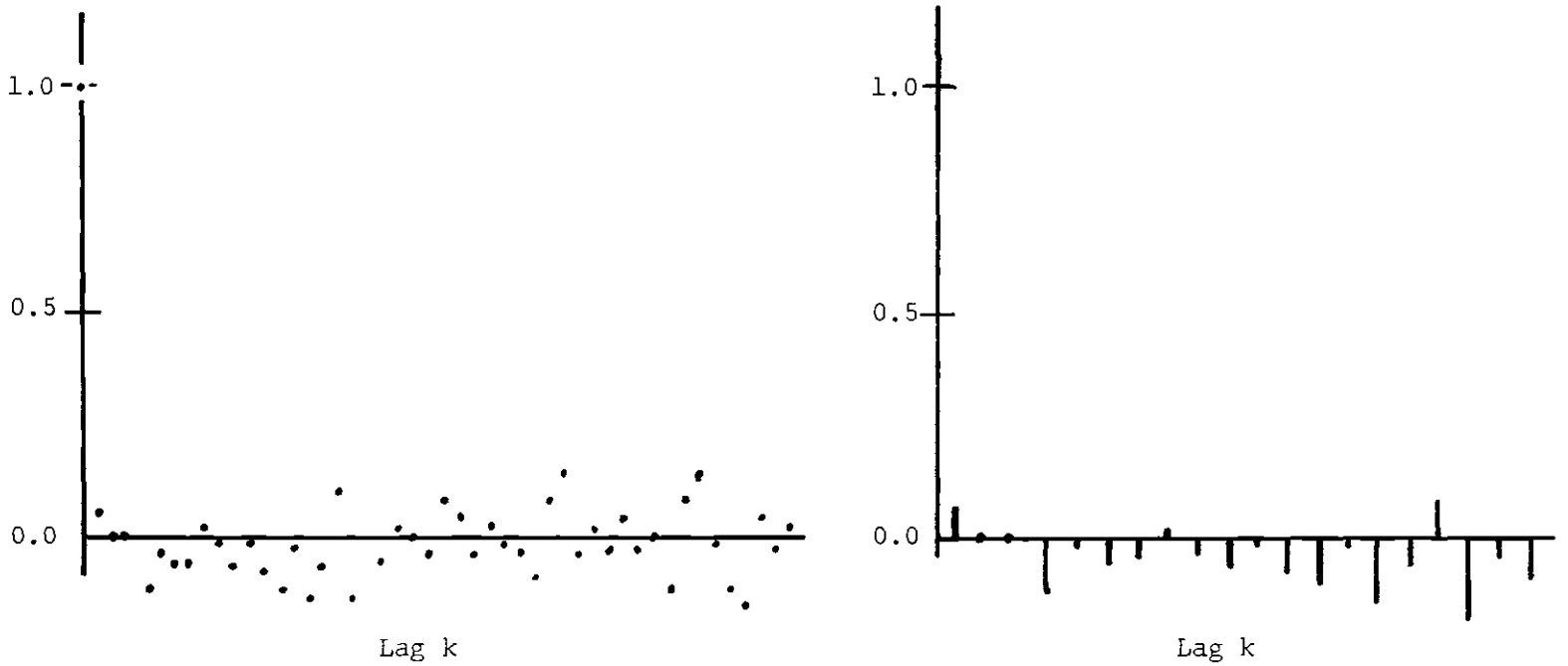
The first observation that should be made is that the different models produced very similar plots for the two links. This indicates that the form of the statistical model used to estimate delay in one model could be fitted to the observed delay found in a different model



a. Sample Autocorrelation Function

b. Partial Autocorrelation Function

Figure 20. Sample Autocorrelation Function and Partial Autocorrelation Function for North Approach Link



a. Sample Autocorrelation Function

b. Partial Autocorrelation Function

Figure 21. Sample Autocorrelation Function and Partial Autocorrelation Function for South Approach Link

by using different estimates for the parameters of the statistical model. That is to say, all models seem to exhibit the same interrelationships in period-by-period or cycle-by-cycle delay.

The second observation is that the two links exhibit different autocorrelation functions. The North approach plot suggests that the delay time during period t has a positive correlation with period t - 1 and all other relationships are zero. The South approach plots suggest independence between each period. Although the link flow volumes for these two links are about the same, a noticeable difference exists in the turning percentages. Five per cent of the vehicles on the North approach will turn left, while no left turns are permitted on the South approach. The existence of left-turn delays appears to be sufficient to cause the appearance of a first order autocorrelation.

Computer Requirements. The program execution time was recorded for each of the test runs and is given in Table 11. The ZONE model was the fastest in all cases. The NEXT model was very fast for simple intersection work but quickly lost any computational advantage it possessed as the network complexity increased. The UB model was the slowest of the models tested. Although the vehicle movement concepts in the UB model are not as complex as the car-following restrictions of the CONT model, the repeated checking of each block and the complex routines for intersection movement require greater computer time to process. The algebraic formulas needed for car-following restrictions are well suited for computer programming.

The computer memory requirements of the different models is important if large networks are being considered. The actual requirements

Table 11. Program Execution Time for Test Runs (in Seconds)

Configuration	Flow	MODEL			
		CONT	UB	ZONE	NEXT
Intersection	Light	17.0 18.0	38.4 38.1	3.4 3.3	6.0 6.0
	Medium	26.0 26.0	47.0 48.1	4.3 4.2	9.0 9.0
	Heavy	49.0 53.0	66.2 78.4	4.3 5.0	25.0 16.2
Arterial	Light	77.2 81.2	139.2 136.0	14.0 15.0	32.4 33.1
	Medium	120.4 114.0	182.5 170.5	20.0 13.4	76.4 76.2
	Heavy	202.0 179.1	289.0 267.0	17.0 18.0	168.1 163.4
Network	Light	221.1 207.0	340.0 355.2	36.0 35.5	199.3 166.0
	Medium	334.3 316.0	462.0 482.0	42.4 36.0	333.0 318.0
	Heavy	577.0 483.0	729.0 923.0	46.4 46.1	861.3 656.0

can be estimated by using the information given in Table 12.

Table 12. Computer Memory Requirements

Model	Base Requirement	Per Node	Per Signal	Per Vehicle	Per Link
CONT	33,000	15	15	12	33
UB*	26,000	17	15	9	26
ZONE**	17,000	12	15	-	15
NEXT	20,500	15	15	8	37

Additional Requirements

*The UB model requires one word of computer memory for each unit block.

**The ZONE model requires one word of computer memory for each zone.

In order to reduce the amount of computer memory necessary, an attempt was made to code information as illustrated in Figure 22.

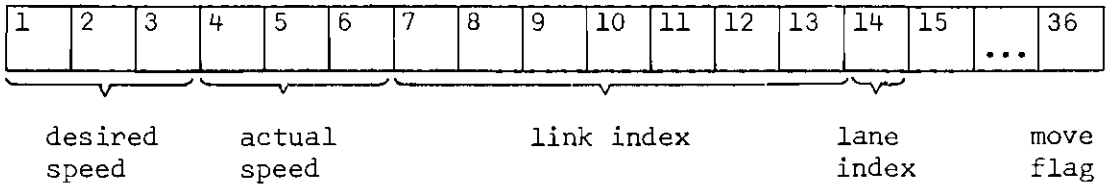


Figure 22. Basic Concept of Coded Data Words

The UB model was selected. Vehicle data were compressed to three computer words and unit blocks were packed three to a word. However, resulting test runs increased running time by a factor of six and the coded model was abandoned.

Secondary Considerations

In addition to the analysis of the effects of modeling strategies, a limited number of secondary problems were investigated.

Scan Time Effects

Zone models reported in the literature (16,33,39) have used scan times of 2, 4 and 5 seconds. To test the effect of scan time on vehicle delay, a series of test runs of the ZONE model was conducted with scan times of 2, 3, 4 and 5 seconds. Since the basic modeling concept was introduced for the purpose of modeling a system larger than a single intersection, the arterial configuration was selected as the test network. Five replications of light, medium and heavy flow rates were run for each scan time. The average system delay per vehicle per link encountered was computed for each run, and the results are presented in Table 13. A two-way analysis of variance was performed on these data, and the results are given in Table 14. Both main effects and their interaction are highly significant. A graphic illustration of these effects is given in Figure 23. To further analyze the interaction of these two factors the Duncan Multiple Range test was performed on the data by individually testing each flow condition. At the low level no significant differences were exhibited between scan time results. At the medium flow rate, two pairs of predictions were significant at a 5% level of confidence: the 3 second-2 second difference and the 3 second-4 second difference. At the heavy flow rate, all pairs of differences were significant except the 2 second-4 second combination.

Table 13. Effects of Scan Time on ZONE Model Response Predictions

Flow	Time Step (Seconds)			
	2	3	4	5
Low	12.74	14.91	12.95	14.50
	13.05	15.91	13.92	15.24
	13.25	15.85	13.22	14.88
	13.38	15.65	13.56	15.07
	13.74	16.65	13.10	14.86
Medium	16.15	24.84	16.24	18.96
	16.46	29.25	16.55	21.98
	16.22	25.99	16.38	23.39
	16.71	24.33	16.01	20.11
	16.67	31.45	16.94	21.40
Heavy	26.34	76.63	23.60	44.43
	28.74	71.60	28.67	50.88
	25.15	81.37	24.28	69.93
	23.60	74.93	21.59	56.69
	31.92	91.63	29.48	78.45

Table 14. Analysis of Variance of the Effects of Scan Time on the ZONE Model

Source of Variance	Sum of Squares	df	Mean Square
Scan Times (T_i)	5268.66	3	1756.22**
Flows (F_j)	12911.04	2	6455.52**
T × F	5449.60	6	908.27**
Error	1159.07	48	24.15
TOTAL	24788.37	59	
**Significant at 1%.			

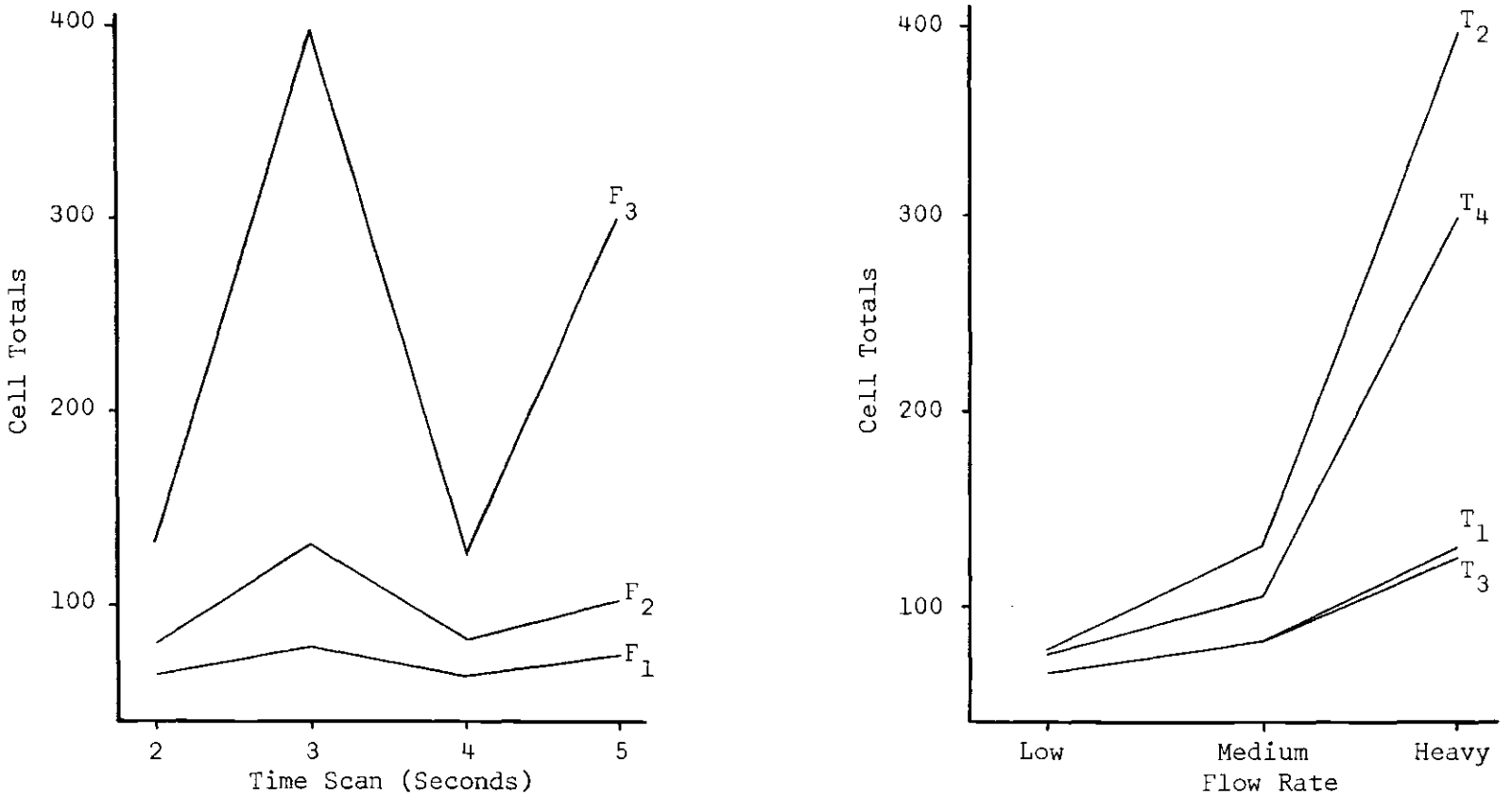


Figure 23. Scan Time-Flow Interaction for ZONE Model

The queue discharge mechanism is thought to be the underlying cause of this behavior. The number of vehicles which can depart a queue is the scan time divided by two seconds per vehicle. Since only an integer number of vehicles may depart the queue, the two- and three-second scan times permit one departure each scan while the four- and five-second scan times permit two discharges. The time between departures is then equal to the desired two seconds for the two- and four-second scan times. The three-second scan is forced to a three-second time between departures, while the average time between departures for the five-second scan is 2-1/2 seconds. The longer departure time yields more queue congestion.

The NEXT model was also tested for step size effects. The scale factor which had been used to force integer clock units was relaxed and the model was permitted to assume values in tenths of seconds. Five replications of three arterial flow rates were submitted for each step size. The data is given in Table 15. Table 16 is the analysis of variance of the data.

Both main effects are highly significant. The interaction between clock units and flows is also significant. Figure 24 illustrates this interaction.

The queue discharge process is again considered to be a contributing factor to this interaction. The smaller clock units allow a closer approximation of the discharge times presented in the model. Integer clock units cause the model to discard the fractional part of the time between discharges, thus causing a more rapid discharge

Table 15. Effects of Clock Units on NEXT Model Response Predictions

Flow	Clock Unit (Seconds)	
	1	1/10
Low	11.09	11.60
	12.08	12.73
	11.39	12.16
	11.34	11.78
	12.97	13.44
Medium	13.36	13.72
	15.02	15.23
	13.06	13.65
	12.97	13.78
	13.66	14.18
Heavy	19.53	29.89
	19.57	32.08
	17.26	20.98
	15.76	19.26
	22.08	25.84

Table 16. Analysis of Variance of Clock Unit Effects on NEXT Model Response Predictions

Source of Variance	Sum of Squares	df	Mean Square
Clock Units (C_i)	51.17	1	51.17**
Flows (F_j)	588.50	2	294.25**
C × F	64.84	2	32.42*
Error	154.56	24	6.44
TOTAL	859.07	29	

*Significant at 5%.
**Significant at 1%.

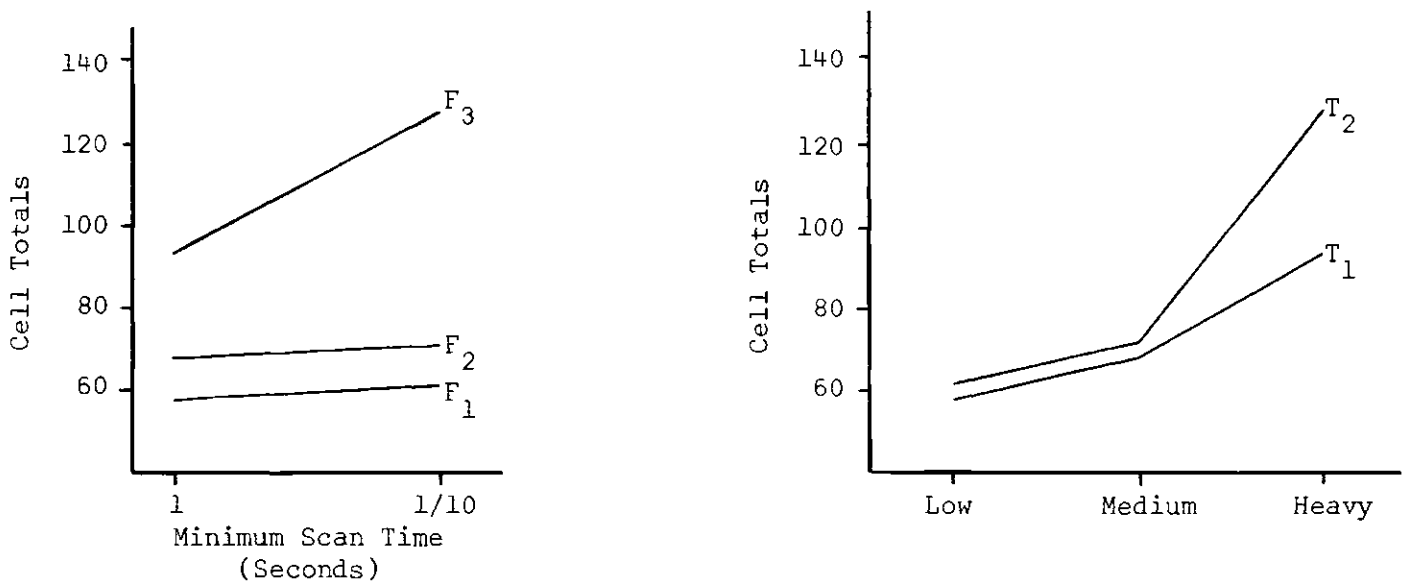


Figure 24. Interaction Between Clock Unit and Flow for NEXT Model

process. As flow increases and the average queue lengths increase, a reduction in delay time is seen.

Network Preloading

Subroutines were written for the CONT and UB models which would calculate the expected internal flow rates for each link of the network. Vehicles were then created and placed on the links before the simulation started. The models were then allowed a five-minute transition time to remove any effects of this starting condition. The running time of the CONT model was 83 seconds faster than the average time indicated for the heavy network configuration. The preloaded UB model bettered the previous results by 123 seconds. Although these results are within the range of random error, it was noted that preload calculations consumed only 3.5 seconds of computer time. Thus it is advisable to consider preloading, if large networks are simulated with the more complex models.

Interarrival Time Distributions

Since much of the literature on traffic simulation is concerned with interarrival distributions, it was decided to test the effects of the assumed distribution. The NEXT model was revised to permit the simulation of interarrival times which followed an Erlang distribution with the identical mean and with k equal to two. Five replicatins of two configurations and three flow volumes were included in the test.

The average delay per vehicle at the main intersection is reported in Table 17. The analysis of variance of these data is given in Table 18.

Table 17. Effects of Interarrival Time Distribution on the NEXT Model Response Distribution

Configuration	Flow	Interarrival Time Distribution	
		Erlang	Exponential
Intersection	Low	16.03 15.70 16.02 15.99 16.93	17.07 16.95 16.41 16.80 17.22
	Medium	18.55 18.26 17.65 18.03 20.54	17.85 18.85 18.68 18.24 19.66
	High	23.68 25.02 24.85 28.70 26.72	28.60 24.59 27.03 24.64 24.89
Artery	Low	11.06 12.34 11.29 11.58 12.18	11.09 12.08 11.39 11.34 12.97
	Medium	13.08 12.76 13.38 12.44 13.20	13.36 15.02 13.06 12.97 13.66
	High	17.39 17.05 16.66 19.72 23.37	19.53 19.57 17.26 15.76 22.08

Table 18. Analysis of Variance of Effects of Interarrival Time Distribution on NEXT Model

Source of Variance	Sum of Squares	df	Mean Square
Distributions (D_i)	1.17	1	1.17
Networks (N_j)	490.09	1	490.09**
Flows (F_k)	746.47	2	373.24**
D × N	.02	1	.02
D × F	.33	2	.16
N × F	13.69	2	6.84*
D × N × F	1.01	2	.51
Error	98.22	48	2.05
TOTAL	1351.01	59	

*Significant at 5%.
**Significant at 1%.

The analysis of variance indicates that there is no significant effect in model performance caused by changing the arrival time distribution. Flow and configuration are highly significant. It is also noted that the interaction between flow and configuration is significant. This interaction was not significant in the analysis of variance for all four models. The use of five replications per cell may have unmasked an unknown interaction.

CHAPTER V

CONCLUSIONS AND RECOMMENDATIONS

Summary

Four computer programs representing different strategies for simulating urban vehicular flow were programmed and tested. A simple two-lane, right-angled intersection was simulated under combinations of varying flow rates and turning probabilities and the resulting delay prediction compared to Webster's equation for average delay.

A series of tests were then conducted to determine the significance of model type, network configuration and flow rate on the prediction of average vehicle delay. A second set of tests were performed to examine the dynamic properties of delay predictions over extended periods of time.

A number of secondary considerations in traffic simulation were examined. First two of the models were selected and the effects of simulation scanning time were examined. Next, a method of preloading the network with vehicles was devised and the reduction in computation time of preloading was compared to the times obtained by starting the networks in an empty state. Finally, one of the models was selected to test the significance of the interarrival time distribution. Model performance using exponentially distributed interarrival times was compared to performance under the assumption of an Erlang distribution.

Conclusions

Effects of Modeling Philosophies on Performance

There is a statistically significant difference in the delay predictions of the models. Although the time dependent sequence of delays obtained from the various models appears to follow the same pattern, the magnitudes of the delay predicted by the models differs significantly.

Model running times also exhibit a dependence on model type. The ZONE model times were 10 to 15 times faster than the more complex UB and CONT models. Differences in computer memory requirements were not as significant as differences in running speeds, and it was found that this parameter was highly dependent on the actual network being modeled.

Warm-Up Time and Preloading Considerations

The results regarding warm-up time were inconclusive. The correlation between delays observed in different time periods appears to be restricted to a one-period (signal cycle time) lag. However, it is noted that this factor depends on turning probabilities and further research is needed to examine other contributing factors.

The magnitude of the savings in computational time of preloading a network with vehicles was not statistically significant, based on the limited sample size. However, the short time necessary to preload the network indicates that a saving should result from preloading larger networks.

Desirable Step Size

Both the step size of the ZONE model and the unit size of the clock for the NEXT model were shown to have a significant effect on delay prediction. Since field data from actual network delays were not available, no specific step size was singled out as being best. Extensive field validation is necessary to fix this parameter.

Interarrival Time Distribution

No significant effect was determined in the choice of an interarrival time distribution. The computational complexity of the logarithmic calculations in the exponential distribution and the Erlang distribution suggest that possibly some other distribution might be more desirable.

Recommendations

Direction of Future Traffic Models

This research has only indicated that a difference exists in the responses produced by different model types. An extensive field validation procedure is necessary before the best model can be identified. This research is greatly needed due to the increasing number of applications of traffic simulation.

The unit block type models do not appear to offer any significant advantages. Large computer memory requirements, slow running speeds and complex programming routines suggest that they should be removed from serious considerations for network modeling.

The tremendous speed advantage of the ZONE model suggests that further research in this modeling technique should be conducted to

improve the performance prediction capabilities of this model. Care must be taken in establishing the desirable step size.

It is the conjecture of this researcher that the NEXT model can be improved to obtain the speed advantage in network situations which was exhibited in the intersection studies. Irregardless, the chain structure used to control infrequent events, such as signal changes, should be employed by all model types. Repeat checking of the status of a green light can only lead to wasted computer processing time. The importance of this factor increases as smaller step sizes are considered.

Model Validation Practices

There is no substitute for the validation of model response to field observations. This validation must contrast the prediction of the performance measures with actual system characteristics under similar situations. If the models are to be used to measure network characteristics, then actual networks must be used for field observations.

Before field validation begins, the method of gathering field data needs to be reviewed and adjusted for network consideration. Methods must be devised for measuring the performance measures in the field as well as in the simulation models. Present techniques are lacking in this area.

Performance Measures. Some research is needed in the area of performance measure definition and identification. In many cases, models have been developed and the performance measures readily available used to contrast system performance. Significant measures need to be identified and then operationally defined.

The measurement and analysis of simulation responses requires more investigation. The simple application of autocorrelation analysis presented in Chapter 4 is but one of a multitude of analytical techniques available to examine the time dependent system responses. Extensive research is needed in this area to analyze both the system and the simulation responses.

APPENDICES

APPENDIX A

INPUT FORMAT SPECIFICATIONS

Each input deck consists of a run specification card and a network data deck. The network data deck has the same format for all four models. The run specification card is the same for the CONT, UB and NEXT models. One additional piece of information is required for the ZONE model.

Run Specification Card

Each input deck must begin with a run specification card. A free field format is used for this card and all quantities are read as integer variables. For the CONT, UB and NEXT models the following data must be entered:

1. Run Index
2. Network Index
3. Amber Time (Seconds)
4. Random Number Seed
5. Data Running Time (Minutes)
6. Warm-Up Time (Minutes)

The ZONE model requires:

1. Run Index
2. Network Index
3. Amber Time (Seconds)
4. Random Number Seed
5. Scan Time Step Size (Seconds)
6. Data Running Time (Minutes)
7. Warm-Up Time (Minutes)

Network Data Deck

The network data deck is divided into three sections. First the data cards on all external nodes are read. Next, pairs of cards are read for each internal node. The first card specifies data characteristics of the node while the second card identifies the signal characteristics. Finally, the data cards on all links are read. Each section of cards is followed by a blank card as a section delimitor.

Each external node in the network must be identified by one card as specified below:

Column	
1	The card identification code 1 must be entered (I1).
2-4	The node identification number must be entered (I3).
5-6	The node type must be entered (I2). 0 - Both generate and terminate. 1 - Generate only. 2 - Terminate only.
10-14	The node generation volume in vehicles per hour (F5.0).
20	The number of lanes per link leading from the node (I1).

Each internal node will have two data cards associated with it. The first card identifies the node characteristics and the format is:

Column	
1	The card identification code 2 must be entered (I1).
2-4	The node identification number must be entered (I3).
5-6	The node type must be entered (I2). 5 - Fixed time signal controlled.
7-9 *10-12* *13-15* *16-18*	The identification number of the nodes connected to the node. A counter-clockwise sequence is assumed starting with one of the major street approach nodes.

19 The node geometry (I1)
1 - Two-lane by two-lane.
3 - Four-lane by four-lane.

The second internal node data card identifies signal control parameters and is specified as follows:

Column

1 The card identification code 3 must be entered (I1).

2-4 The node identification number must be entered (I3).

5-7 The signal cycle time in seconds (I3).

8-10 The major street offset time* in seconds (I3).

11-13 The major street green time in seconds (I3).

14-16 The minor street offset time in seconds (I3).

17-19 The minor street green time in seconds (I3).

Each network link requires one link data card in the network data deck. The format specification is given below:

Column

1 The card identification code 4 must be entered (I1).

2-4 The identification number of the source node must be entered (I3).

5-7 The identification number of the head node must be entered (I3).

8-11 The link length in feet (I4).

12 The number of lanes (I1).

13-15 The source node identification number of the link opposing left-turning traffic (I3).

16-18 The probability of a left turn (in parts per thousand) at the head node (F3.3).

19-21 The probability of a straight movement at the head node (F3.3).

22-24
25-27 The head node identification numbers of the destination links for left-, straighting and right-turning traffic.
28-30

---

*Here offset time is defined to be the number of seconds until the first red phase.

APPENDIX B

FORTRAN LISTING OF CONT MODEL

```
DEFINS PROCEDURE
C
C      ..... SETUP FOR NODES.
C
       PARAMETER MAXNO=35
       REAL NOPARS,NOCLK
       COMMON /NODES/ NOIDNO(MAXNO),NOTYPE(MAXNO),NONXT(MAXNO,4),
      1  NOPARS(MAXNO,2),NOINP(MAXNO,4),NOOUTP(MAXNO),
      2  NOSEED(MAXNO),NOSIG(MAXNO),NOCLK(MAXNO),NUMNOS
C
C      ..... SETUP FOR SIGNALS.
C
       PARAMETER MAXSIG=20
       INTEGER AMBERT,SIGINP,SIGCYC,SIGSTA,SIGOFF,SIGGRN,SIGRED,SIGCLK
       COMMON /SIG/ SIGCYC(MAXSIG),SIGSTA(MAXSIG,2),SIGOFF(MAXSIG,2),
      1  SIGGRN(MAXSIG,2),SIGRED(MAXSIG,2),SIGCLK(MAXSIG,2),
      2  SIGINP(MAXSIG,4),NUMSIG,AMBERT
C
C      ..... SETUP FOR LINKS.
C
       PARAMETER MAXLKS=100
       REAL LKPROB
       COMMON /LINKS/ LKID(MAXLKS,2),LKLNG(MAXLKS),LKLANS(MAXLKS),
      1  LKNOD(MAXLKS),LKOPOS(MAXLKS),LKLEAD(MAXLKS,2),LKLAST(MAXLKS,2),
      2  LKHOLD(MAXLKS,2),LKPROB(MAXLKS,2),LKDEST(MAXLKS,3),NUMLKS,
      3  LKTAG(MAXLKS,2),LKDEL(MAXLKS,2),LKDFLD(MAXLKS,3),LKVL(MAXLKS,2),
      4  LKVD(MAXLKS,3),LKSTOP(MAXLKS,2),LKMAXQ(MAXLKS,2)
C
C      ..... SETUP FOR VEHICLES.
C
       PARAMETER MAXVEH=2000
       INTEGER VLANE,VLKTIM,VLK,VLEAD,VFOLOW,VTURN,VTAG,VACC
       COMMON /VEH/ VDSP(MAXVEH),VASP(MAXVEH),VACC(MAXVEH),VLK(MAXVEH),
      1  VPOS(MAXVEH),VLEAD(MAXVEH),VFOLOW(MAXVEH),VTURN(MAXVEH),
      2  VLKTIM(MAXVEH),VLANE(MAXVEH),VTAG(MAXVEH),VHOLD(MAXVEH),
      3  NUMVEH
C
C      ..... GENERAL COMMON VARIABLES.
C
       INTEGER CLOCK,FINISH,WARMUP,DELT
       COMMON CLOCK,ISEED,FINISH,WARMUP,DELT
C
C      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C      *                                                         *
C      *      DEFINITION OF VARIABLES USED IN THE PROGRAM.       *
C      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C      *                                                         *
C      *                                                         *
C      *                                                         *
C      *      AMBERT - AMBER TIME IN SECONDS.
C      *      CLOCK - MASTER SIMULATION CLOCK.                   *
C      *      DELT - DELTA TIME, THE TIME STEP.
C      *      FINISH - THE ABSOLUTE STOPPING TIME.
C      *      ISEED - THE INITIAL RANDOM NUMBER SEED.
C      *      ITH - INDEX USED TO POINT TO SPECIFIC SIGNAL.      *
C      *      KTH - INDEX USED TO POINT TO SPECIFIC LINK.        *
```

```
C     *     LKDEL(KTH,1-2) - CUMULATIVE DELAY ON LINK BY LANES.
C     *     LKDELD(KTH,1-3) - CUMULATIVE DELAY ON LINK BY DESTINATION.
C     *     LKDEST(KTH,1-3) - LINK DESTINATIONS: LEFT,STR,RIGHT.  *
C     *     LKHOLD(KTH,1-2) - INDEX TO VEHICLE IN HOLDING AREA    *
C     *     LKID(KTH,1-2) - SYMBOLIC I.D. OF TAIL(1) AND HEAD(2). *
C     *     LKLANS(KTH) - NUMBER OF LANES.                        *
C     *     LKLAST(KTH,1-2) - INDEX TO LAST VEHICLE               *
C     *     LKLEAD(KTH,1-2) - INDEX TO LEADER                     *
C     *     LKLNG(KTH) - LINK LENGTH (IN FEET  ).                 *
C     *     LKMAXQ(KTH,1-2) - MAXIMUM QUEUE LENGTH, BY LANES.
C     *     LKNOD(KTH) - INDEX TO NODE AT HEAD OF LINK.           *
C     *     LKOPOS(KTH) - LINK APPOSING A LEFT TURN.              *
C     *     LKPROB(KTH,1-2) - CUMULATIVE PROB OF LEFT & STR. MOVE.*
C     *     LKSTOP(KTH,1-2) - TOTAL NUMBER OF STOPS ON LINK BY LANES.
C     *     LKTAG(KTH,1-2) - LINK TAG INDICATOR :                 *
C     *          0 - OK,  1 - MUST TAG A VEHICLE TO STOP.          *
C     *     LKVD(KTH,1-3) - VEHICLE COUNT BY DESTINATION.
C     *     LKVL(KTH,1-2) - VEHICLE COUNT BY LANES.
C     *     MAXLKS - MAXIMUM POSSIBLE NUMBER OF LINKS.            *
C     *     MAXNO  - MAXIMUM POSSIBLE NUMBER OF NODES.            *
C     *     MAXSIG - MAXIMUM POSSIBLE NUMBER OF SIGNALS.          *
C     *     MAXVEH - MAXIMUM POSSIBLE NUMBER OF VEHICLES.         *
C     *     NOCLK(NTH) - TIME OF NEXT GENERATION FROM NODE.
C     *     NOIDNO(NTH) - SYMBOLIC NODE IDENTIFIER.               *
C     *     NOINP(NTH,1-4) - INDEX TO LINKS INPUTTING TO THIS NODE*
C     *     NONXT(NTH,1-4) - SYMBOLIC IDENTIFIERS OF ADJ. NODES.  *
C     *     NOOUTP(NTH) - INDEX TO LINKS TAKING FROM THIS NODE    *
C     *     NOPARS(NTH,1-2) - PARAMETERS FOR GENERATE NODES.      *
C     *     NOSIG(NTH) - INDEX TO SIGNAL FOR NTH NODE.            *
C     *     NOTYPE(NTH) - NODE TYPE:                              *
C     *                       0 - BOTH GENERATE AND TERMINATE,    *
C     *                       1 - GENERATE ONLY,                  *
C     *                       2 - TERMINATE ONLY,                 *
C     *                       4 - TWO WAY STOP,                   *
C     *                       5 - FIXED TIME CONTROLLED.          *
C     *     NTH - INDEX USED TO POINT TO SPECIFIC NODE.           *
C     *     NUMLKS - NUMBER OF LINKS IN MODEL.                    *
C     *     NUMNOS - NUMBER OF NODES.                             *
C     *     NUMSIG - NUMBER OF SIGNALS.                           *
C     *     NUMVEH - NUMBER OF VEHICLES IN NETWORK.               *
C     *     SIGCLK(ITH,1-2) - TIME OF NEXT STATE CHANGE TO SIGNAL.
C     *     SIGCYC(ITH) - CYCLE LENGTH OF ITH SIGNAL.             *
C     *     SIGGRN(ITH,1-2) - LENGTH OF GREEN SIGNAL
C     *     SIGINP(ITH,1-4) - LINKS INPUTTING TO THE ITH SIGNAL.
C     *     SIGOFF(ITH,1-2) - OFFSET TIME.
C     *     SIGRED(ITH,1-2) - RED TIME FOR ITH SIGNAL.
C     *     SIGSTA(ITH,1-2) - SIGNAL STATE:
C     *             0 - GREEN.
C     *             1 - AMBER,
C     *             2 - RED.
C     *     VACC(JTH) - ACCELERATION INDICATOR FOR LEFT TURNS     *
C     *     VASP(JTH) - VEHICLE'S ACTUAL SPEED.                   *
C     *     VDSP(JTH) - VEHICLE'S DESIRED SPEED.                  *
C     *     VFOLOW(JTH) - VEHICLE FOLLOWING THE JTH VEHICLE       *
C     *     VHOLD(JTH) - NUMBER OF FEET MOVED BEFORE HOLDING      *
C     *     VLANE(JTH) - LANE DESIRED.                            *
C     *     VLEAD(JTH) - VEHICLE LEADING THE JTH VEHICLE.         *
```

```
C     *     VLK(JTH) - INDEX OF CURRENT OR MOST RECENT LINK.      *
C     *     VLKTIM(JTH) CLOCK TIME VEHICLE SHOULD LEAVE LINK.
C     *     VPOS(JTH) - POSITION OF JTH VEHICLE FROM END OF LINK  *
C     *     VTAG(JTH) - STOPPING TAG INDICATOR,  1 - STOP.        *
C     *     VTURN(JTH) - TURN INDICATOR:                          *
C     *                          1 - LEFT,                        *
C     *                          2 - STRAIGHT,                    *
C     *                          3 - RIGHT.                       *
C     *     WARMUP - THE AMOUNT OF TIME FOR WARMUP.
C     *                                                           *
C     *                                                           *
C     *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C     *                                                           *
C     *                                                           *
C     *                                                           *
C     *                                                           *
C     *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C
C
C
 END
C
C     ..... THIS IS THE MAIN PROGRAM FOR THE CONTINUOUS MODEL.
C           CLOCK IN SECONDS.
C
C
      INCLUDE DEFINS
C
C     ..... INITIALIZATION
      NTH = 0
      DELT = 1
      CLOCK = -DELT
C
C     ..... RUN PARAMETERS
C
      READ(5,1) IRUN,INET,AMBERT,ISEED,FINISH,WARMUP
    1 FORMAT( )
      MODEL = 'CONT'
      WRITE(21) MODEL,IRUN,INET,ISEED
      WRITE(22) MODEL,IRUN,INET,ISEED
      PUNCH 6,IRUN,INET,ISEED
    6 FORMAT('CONT MODEL  IRUN = ',I4,'  INET = ',I4,' ISEED = ',I12)
      WARMUP = WARMUP * 60
      FINISH = FINISH * 60 + WARMUP
      WRITE(6,2) IRUN,INET,ISEED,FINISH,WARMUP
    2 FORMAT('1**************** CONTINUOUS MODEL ****************',//
     1  ' RUN NUMBER ',I5,//,'   NETWORK ',I5,'    ISEED ',I9,
     2  //,'   FINISH = ',I8,' SECONDS.  WARMUP =',I4,' SECONDS.',//)
C
C     ..... CALL INPUT SUBROUTINES.
C
      CALL EXTERN(NTH)
      CALL INTERN(NTH)
      CALL INPLKS
C
C     ..... CALL SETUP TO FINISH DATA MANIPULATION.
C
```

```
      CALL SETUP
      CALL DUMP(0,'DATA C',',HECK  ',n)
C
C     ..... THE SIMULATION.
C
 1000 CLOCK = CLOCK + DELT
      IF(MOD(CLOCK,90).EQ.0)  CALL STATA(2)
      WRITE(22) CLOCK,NUMVEH
      IF(CLOCK .GE. WARMUP) GO TO 2000
      CALL SIGCHK
      CALL VEGEN
      CALL LKMOVE
      CALL HOLDOU
      GO TO 1000
C
C     ..... WARMUP OVER, CLEAR STATISTICS.
C
 2000 CALL STAT(1)
      CALL STATA(1)
      GO TO 2200
 2100 CLOCK = CLOCK + DELT
      IF(MOD(CLOCK,90).EQ.0)  CALL STATA(2)
      WRITE(22) CLOCK,NUMVEH
      IF(CLOCK .GE. FINISH) GO TO 3000
 2200 CALL SIGCHK
      CALL VEGEN
      CALL LKMOVE
      CALL HOLDOU
      GO TO 2100
C
C     ..... SIMULATION OVER, PRINT STATISTICS.
C
 3000 CALL STAT(2)
      ENDFILE 21
      ENDFILE 22
      STOP
      END
      SUBROUTINE DUMP(M,N1,N2,K)
C
C     ..... THIS SUBROUTINE PROVIDES A DUMP OF THE VARIABLES
C           IN CASE OF AN ERROR.  THE VALUE OF K DETERMINES
C           THE TYPE OF DUMP:
C                              0 - SET UP CHECK, CALL RETURN
C                                  AFTER PRINTOUT ON NODES, ETC.
C                              1 - ERROR ON INPUT,
C                            > 1 - ERROR WHILE RUNNING - LINE NUMBER.
C
      INCLUDE DEFINS
C
C     ..... GENERAL INFORMATION DUMP.
C
      WRITE(6,1) K
1     FORMAT(///' ERROR DUMP OUTPUT.  K =',I5)
  100 WRITE(6,3) M,N1,N2
3     FORMAT('   THING ',I5,' NOW BEING PROCESSED.',/,' MESSAGE ='
     1  ,1X,2A6)
C
```

```
C      ..... NODE OUTPUT.
C
 1001 MAX = MAXNO
      WRITE(6,1010) MAX,NUMNOS
 1010 FORMAT(///,' NODE OUTPUT          MAX = ',I2,12X,
     1  'NUMBER = ',I2,//,' INDEX    ID    NEXT NODE ID',
     2  '    TYPE   INDEX LINKS IN    LINK OUT   SIGNAL',
     3  '    CLOCK  PARAMETERS',//)
      DO 101 I=1,NUMNOS
  101 WRITE(6,1011) I,NOIDNO(I),(NONXT(I,K),K=1,4),NOTYPE(I),
     1  (NOINP(I,K),K=1,4),NOOUTP(I),NOSIG(I),NOCLK(I),
     2  (NOPARS(I,K),K=1,2)
 1011 FORMAT(I4,6X,I2,3X,4I3,6X,I2,2X,4I4,7X,I2,8X,I2,5X,
     1  F5.0,1X,F5.0,1X,F5.0)
C
C      ..... SIGNAL OUTPUT
C
      MAX = MAXSIG
      LAM = AMBERT
      WRITE(6,1020) MAX,NUMSIG,LAM
 1020 FORMAT(///,' SIGNAL OUTPUT       MAX = ',I2,'       NUMBER = ',
     1  I2,'       AMBER TIME = ',I2,//,23X,'* * * *   MAJOR',
     2  '   * * * *',22X,'* * * *   MINOR   * * * *',/,
     3  ' INDEX  CYCLE',2(6X,'OFFSET  GREEN   RED    STATE CLOCK   ',
     4  ))
      DO 102 I=1,NUMSIG
  102 WRITE(6,1021) I,SIGCYC(I),(SIGOFF(I,K),SIGGRN(I,K),SIGRED(I,K),
     1  SIGSTA(I,K),SIGCLK(I,K),K=1,2)
 1021 FORMAT(I4,5X,I3,,2(8X,I3,5X,I3,4X,I3,5X,I2,3X,I5,3X))
C
C      ..... LINK OUTPUT.
C
      MAX = MAXLKS
      WRITE(6,1030) MAX,NUMLKS
 1030 FORMAT(///,' LINK OUTPUT      MAX = ',I3,'     NUMBER = ',I3,//,
     1  ' INDEX    ID No.  LENGTH  LANES   NODE    LK OPOS    LK',
     2  ' DESTINATIONS    PROBABILITIES  LEAD  LEAD  LAST  LAST  TAG',
     3  '  HOLD AREA',//)
      DO 103 I=1,NUMLKS
  103 WRITE(6,1031) I,LKID(I,1),LKID(I,2),LKLNG(I),LKLANS(I),LKNOD(I),
     1  LKOPOS(I),(LKDEST(I,K),K=1,3),(LKPROB(I,K),K=1,2),
     2  LKLEAD(I,1),LKLEAD(I,2),LKLAST(I,1),LKLAST(I,2),LKTAG(I,1),
     3  LKTAG(I,2),LKHOLD(I,1),LKHOLD(I,2)
 1031 FORMAT(I5,4X,2I3,I7,6X,I1,6X,I2,7X,I3,3X,2I6,3X,I3,2X,
     1  2F7.3,3X,4(I4,2X),I1,1X,I1,2X,I4,2X,I4)
      IF(K .EQ. 0)  RETURN
      IF(K .EQ. 1) STOP
C
C      ..... VEHICLE OUTPUT.
C
      MAX = MAXVEH
      WRITE(6,1050) MAX,NUMVEH
 1050 FORMAT(///,' VEHICLE OUTPUT    MAX = ',I4,'   NUMBER = ',I4,//,
     1  ' INDEX   DSP   ASP   LINK   LANE   TURN   POSITION   ACC   TAG'
     2  ,'   LEAD   FOLLOW   HOLD',//)
      DO 105 I=1,MAXVEH
      IF(VDSP(I) .LE. 0)  GO TO 105
```

```
      WRITE(6,1051) I,VNSP(I),VASP(I),VLK(I),VLANE(I),VTURN(I),
     1  VPOS(I),VACC(I),VTAG(I),VLEAD(I),VFOLOW(I),VHOLD(I)
 1051 FORMAT(I4,5X,F3.0,3X,I3,6X,I1,6X,I1,5X,F5.0,6X,I1,5X,I1,
     1  4X,I4,4X,I4,4X,F3.0)
  105 CONTINUE
      STOP
      END
      SUBROUTINE EXTERN(NTH)
C
C     ..... THIS SUBROUTINE READS THE DATA CARDS ON EXTERNAL
C           NODES ONLY.  CONTROL IS RETURNED TO THE CALLING
C           PROGRAM AFTER A BLANK CARD IS FOUND
C
      INCLUDE DEFINS
      INTEGER ALPHA
 1000 NTH=NTH+1
      READ(5,101) ALPHA,NOIDNO(NTH),NOTYPE(NTH),NONXT(NTH,1),
     1  NOPARS(NTH,1),NOPARS(NTH,2)
      IF(ALPHA .EQ. 1H )  GO TO 2000
      GO TO 1000
 2000 NTH=NTH-1
      RETURN
  101 FORMAT(A1,I3,I2,I3,2F5.0)
      END
      SUBROUTINE GAPCHK(KTH,KOUT,ITH,II)
C
C     ..... THIS SUBROUTINE FINDS THE SIZE OF THE GAP ON THE
C           KTH LINK AND TESTS FOR LEFT HAND TURNS. KOUT EQUALS
C           0 IF BIG ENOUGH AND 1 IF GAP TOO SMALL.
C
      INCLUDE DEFINS
      DIMENSION PROB(20)
      DATA PROB / -1., -1., .02, .10, .22, .36, .50, .61, .74,
     1   .81, .84, .90, .92, .95, .96, .97, .98, .99, .995, .9995  /
C
C     ..... IF SIGNAL RED, DO NOT BOTHER WITH GAP CHECKING.
C
      IF(SIGSTA(ITH,II) .EQ. 2)   GO TO 2000
C
C     ..... BRANCH ON NUMBER OF LANES.
C
      IF(LKLANS(KTH) .EQ. 2) GO TO 1300
C
C     ..... SINGLE LANE APPROACH.
C
      JTH = LKLEAD(KTH,1)
 1252 IF(JTH .EQ. 0)   GO TO 2000
      IF(VPOS(JTH) .LE. LKLNG(KTH) + 20)  GO TO 1260
      JTH = VFOLOW(JTH)
      GO TO 1252
 1260 IF(VASP(JTH) .LE. 4.0)   GO TO 2000
      IF(VTURN(JTH) .EQ. 1)   GO TO 2000
      IF(VTAG(JTH) .EQ. 1)  GO TO 2000
      GAP = (LKLNG(KTH) + 20-VPOS(JTH))/VASP(JTH)
      GO TO 1500
C
C     ..... TWO LANE APPROACH, FIND THE SHORTEST GAP.
```

```
C
 1300 GAP1 = 999.
      JTH = LKLEAD(KTH,1)
 1312 IF(JTH .EQ. 0)  GO TO 1400
      IF(VPOS(JTH) .LE. LKLNG(KTH)+20)  GO TO 1320
      JTH = VFOLOW(JTH)
      GO TO 1312
 1320 IF (VASP(JTH) .LE. 4.0)  GO TO 1400
      IF(VTURN(JTH) .EQ. 1)  GO TO 1400
      IF(VTAG(JTH) .EQ. 1)  GO TO 1400
      GAP1 = (LKLNG(KTH) + 20 - VPOS(JTH))/VASP(JTH)
 1400 GAP2 = 999.
      JTH = LKLEAD(KTH,2)
 1412 IF(JTH .EQ. 0)  GO TO 1450
      IF(VPOS(JTH) .LE. LKLNG(KTH)+20)  GO TO 1420
      JTH = VFOLOW(JTH)
      GO TO 1412
 1420 IF(VASP(JTH) .LE. 4.0)  GO TO 1450
      IF(VTAG(JTH) .EQ. 1)  GO TO 1450
      GAP2 = (LKLNG(KTH) + 20 - VPOS(JTH))/VASP(JTH)
 1450 GAP = MIN(GAP1,GAP2)
C
C     ..... TEST THE ACCEPTANCE DISTRIBUTION.
C
 1500 IF(GAP .GE. 20.0)  GO TO 2000
      IF(GAP .LE. 2)  GO TO 1750
      INDEX = GAP + 0.5
      IF(RN1(ISEED) .LE. PROB(INDEX))  GO TO 2000
C
C     ..... REJECT.
 1750 KOUT = 1
      RETURN
C
C     ..... ACCEPT.
 2000 KOUT = 0
      RETURN
      END
      SUBROUTINE HOLDIN(JTH,KTHOLD,KTHNEW,INDEX,DIST)
C
C     ..... REMOVES VEHICLE FROM OLD LINK AND PLACES IT ON NEW.
C
      INCLUDE DEFINS
      INTEGER DELAY
C
C     .....IF NEW VEHICLE, SKIP FIRST SECTION.
C
      IF(KTHOLD .EQ. 0)  GO TO 500
C
C     ..... REMOVE FROM CHAIN ON OLD LINK.
C
      ILANE = VLANE(JTH)
      JTHNXT = VFOLOW(JTH)
      LKLEAD(KTHOLD,ILANE) = JTHNXT
      ITURN = VTURN(JTH)
      DELAY = CLOCK - VLKTIM(JTH)
      WRITE(21) CLOCK,KTHOLD,JTH,DELAY
      LKDEL(KTHOLD,ILANE) = LKDEL(KTHOLD,ILANE) + DELAY
```

```
      LKDELD(KTHOLD,ITURN) = LKDELD(KTHOLD,ITURN) + DELAY
      LKVL(KTHOLD,ILANE) = LKVL(KTHOLD,ILANE) + 1
      LKVD(KTHOLD,ITURN) = LKVD(KTHOLD,ITURN) + 1
      IF(JTHNXT .EQ. 0)  GO TO 100
      VLEAD(JTHNXT) = 0
      VFOLOW(JTH) = 0
      GO TO 500
  100 LKLAST(KTHOLD,ILANE)  = 0
C
C     ..... PLACE IN HOLDING AREA.
C
  500 LKHOLD(KTHNEW,INDEX) = JTH
      VHOLD(JTH) = DIST
C**** WRITE(6,1) JTH,KTHOLD,KTHNEW,INDEX,DIST
    1 FORMAT('   HOLDIN VEH ',I4,' FROM LINK ',I3,' TO LINK ',I3,
     1  I2,' DIST ',F3.0)
      RETURN
      END
      SUBROUTINE HOLDOU
C
C     ..... THIS SUBROUTINE TAKES VEHICLES OUT OF A HOLDING
C           AREA AND MOVES THEM ONTO THE LINK.
C
      INCLUDE DEFINS
      DIMENSION LKFACT(2,3,3)
      DATA LKFACT / 188,232, 30,60, 145,145, 336,397, 30,60, 273,273,
     1              489,568, 30,60, 412,412 /
      DEFINE LFACT(LANE,ITURN,ISPEED) = LKFACT(LANE,ITURN,ISPEED)
C**** WRITE(6,1)
    1 FORMAT(' HOLDOU')
      DO 3000 KTH=1,NUMLKS
      LANE = 1
C
C     ..... BRANCH ON NUMBER OF LANES
      IF(LKLANS(KTH) .EQ. 2)  GO TO 2000
C
C     ..... SINGLE LANE.  LOOK AT HOLDING AREA. IF EMPTY SKIP.
      LANE = 1
      INDEX = 1
      JTH = LKHOLD(KTH,1)
      IF(JTH .EQ. 0)  GO TO 3000
C
C     ..... IF NEW VEHICLE TO THIS LINK, ASSIGN PARAMETERS.
      IF(VLK(JTH) .EQ. KTH)  GO TO 1100
      VLK(JTH) = KTH
      VLANE(JTH) = 1
      X = RN1(ISEED)
      DO 1050 I=1,2
      IF(X .LE. LKPROB(KTH,I))  GO TO 1055
 1050 CONTINUE
      VTURN(JTH) = 3
      GO TO 1075
 1055 VTURN(JTH) = I
 1075 IF(IFIX(VDSP(JTH)) - 54)  1081,1082,1083
 1081 VLKTIM(JTH) = CLOCK + (LKLNG(KTH) + VHOLD(JTH)
     1      + LFACT(1,VTURN(JTH),1) - VDSP(JTH)) / VDSP(JTH)
      GO TO 1100
```

```
 1082 VLKTIM(JTH) = CLOCK + (LKLNG(KTH) + VHOLD(JTH)
     1      + LFACT(1,VTURN(JTH),2) - VDSP(JTH)) / VDSP(JTH)
      GO TO 1100
 1083 VLKTIM(JTH) = CLOCK + (LKLNG(KTH) + VHOLD(JTH)
     1      + LFACT(1,VTURN(JTH),3) - VDSP(JTH)) / VDSP(JTH)
 1100 ZA = ZACCEL(VASP(JTH),VDSP(JTH))
C
C
      JTHCHK = LKLAST(KTH,1)
      IF(JTHCHK .EQ. 0)  GO TO 2900
      ZS = ZSPACE(0.0,VASP(JTH),VPOS(JTHCHK)+VHOLD(JTH),VASP(JTHCHK))
      Z = MIN(ZA,ZS) - VHOLD(JTH)
      IF(Z .LT. 0)  GO TO 2950
      GO TO 2975
C
C
C     ..... TWO LANE LINK PROCESSING. LOOK AT HOLDING AREAS.
C
 2000 INDEX = 1
      JTH = LKHOLD(KTH,1)
      IF(JTH .NE. 0)  GO TO 2100
 2001 INDEX = 2
      JTH = LKHOLD(KTH,2)
      IF(JTH .EQ. 0)  GO TO 3000
 2100 ZA = ZACCEL(VASP(JTH),VDSP(JTH))
C**** WRITE(6,5) ZA,VASP(JTH),VDSP(JTH)
    5 FORMAT()
C
C     ..... FOUND A VEHICLE IN THE HOLDING AREA.  IF VEHICLE
C           NEW TO THIS LINK, ASSIGN PARAMETERS.
C
      IF(VLK(JTH) .EQ. KTH)  GO TO 2250
      VLK(JTH) = KTH
      X = RN1(ISEED)
      IF(X .GT. LKPROB(KTH,1))  GO TO 2140
      VTURN(JTH) = 1
      VLANE(JTH) = 1
      GO TO 2149
 2140 IF(X .GT. LKPROB(KTH,2))  GO TO 2145
      IF(VTURN(JTH) .EQ. 2)  GO TO 2149
      VTURN(JTH) = 2
      VLANE(JTH) = RN1(ISEED) * LKLANS(KTH) + 1
      GO TO 2149
 2145 VTURN(JTH) = 3
      VLANE(JTH) = LKLANS(KTH)
 2149 IF(IFIX(VDSP(JTH)) - 54)  2181,2182,2183
 2181 VLKTIM(JTH) = CLOCK + (LKLNG(KTH) + VHOLD(JTH)
     1      + LFACT(2,VTURN(JTH),1) - VDSP(JTH)) / VDSP(JTH)
      GO TO 2250
 2182 VLKTIM(JTH) = CLOCK + (LKLNG(KTH) + VHOLD(JTH)
     1      + LFACT(2,VTURN(JTH),2) - VDSP(JTH)) / VDSP(JTH)
      GO TO 2250
 2183 VLKTIM(JTH) = CLOCK + (LKLNG(KTH) + VHOLD(JTH)
     1      + LFACT(2,VTURN(JTH),3) - VDSP(JTH)) / VDSP(JTH)
C
C     ..... FIND LEADER IN DESIRED LANE.
C
```

```
 2250 LANE = VLANE(JTH)
      JTHCHK = LKLAST(KTH,LANE)
      IF(JTHCHK .EQ. 0)  GO TO 2900
      ZS = ZSPACE(0.0,VASP(JTH),VPOS(JTHCHK)+VHOLD(JTH),VASP(JTHCHK))
C**** WRITE(6,5) JTH,JTHCHK,ZS,VPOS(JTHCHK),VASP(JTHCHK)
      Z = MIN(ZA,ZS) - VHOLD(JTH)
      IF(ZS .LT. ZA)  VASP(JTH) = ZS
      IF(Z .LT. 0)  GO TO 2950
      GO TO 2975
C
C     ..... ADVANCE ZA,  NO OTHER VEHICLE ON LINK.
C
 2900 ZA = ZA - VHOLD(JTH)
      IF(ZA .LE. 0)  GO TO 2950
      LKHOLD(KTH,INDEX) = 0
      VPOS(JTH) = ZA
      VASP(JTH) = 2*(ZA+VHOLD(JTH)) - VASP(JTH)
      LKLEAD(KTH,LANE) = JTH
      LKLAST(KTH,LANE) = JTH
      VHOLD(JTH) = 0
C**** WRITE(6,2) JTH,KTH,VPOS(JTH),VASP(JTH)
    2 FORMAT('   VEH ',I4,' ON LINK ',I3,' MOVES TO POSITION ',F3.0,
     1  ' SPEED = ',F4.1)
      GO TO 2999
C
C     ..... DONT MOVE FROM HOLDING AREA.
C
 2950 VASP(JTH) = 0
      VHOLD(JTH) = 0
      VPOS(JTH) = 0
C**** WRITE(6,3) JTH,KTH
    3 FORMAT('   VEH ',I4,' STAYS IN HOLDING AREA ON LINK ',I3)
      GO TO 2999
C
C     ..... ADVANCE TO Z.
C
 2975 LKHOLD(KTH,INDEX) = 0
      LKLAST(KTH,LANE) = JTH
      VPOS(JTH) = Z
      VASP(JTH) = 2*(Z+VHOLD(JTH)) - VASP(JTH)
      VLEAD(JTH) = JTHCHK
      VHOLD(JTH) = 0
      VFOLOW(JTHCHK) = JTH
C**** WRITE(6,4) JTH,KTH,VPOS(JTH),JTHCHK
    4 FORMAT('   VEH ',I4,' ON LINK ',I3,' MOVES TO POSITION ',F3.0,
     1  ' BEHIND VEH ',I4)
 2999 IF(INDEX .NE. LKLANS(KTH))  GO TO 2001
 3000 CONTINUE
      RETURN
      END
      FUNCTION IBHOLD(KTH)
C
C     ..... THIS FUNCTION SEARCHES THE HOLDING AREA OF THE
C           KTH LINK.  IF THERE IS AN EMPTY SPACE IN THE
C           HOLDING AREA, THE LANE INDEX IS RETURNED AS
C           THE VALUE OF THE FUNCTION.  IF THE AREA IS FULL,
C           ZERO IS THE RETURNED VALUE OF THE FUNCTION.
```

```
C
      INCLUDE DEFINS
C
C     ..... BRANCH BASED ON THE NUMBER OF LANES.
      IBHOLD = 0
      IF(LKLANS(KTH) .EQ. 2)  GO TO 1000
C
C     ..... SINGLE LANE, IF HOLDING AREA FULL THEN RETURN.
      IF(LKHOLD(KTH,1) .NE. 0)  RETURN
      IBHOLD = 1
      RETURN
C
C     ..... TWO LANES - MUST CHECK BOTH HOLDING AREAS.
C
 1000 IF(LKHOLD(KTH,1) .NE. 0)  GO TO 1010
      IBHOLD = 1
      RETURN
 1010 IF(LKHOLD(KTH,2) .NE. 0)  RETURN
      IBHOLD = 2
      RETURN
      END
      SUBROUTINE INPLKS
C
C     ..... THIS SUBROUTINE READS CARDS FOR ALL NETWORK LINKS.
C           AFTER ALL DATA CARDS HAVE BEEN READ, THE ROUTINE
C           CALCULATES SOME CROSS REFERENCE INDICES.
C
      INCLUDE DEFINS
C
C     ..... THIS IS THE LOCAL DECLARATION.
C
      INTEGER ALPHA
      KTH=0
 1000 KTH=KTH+1
      READ(5,101) ALPHA,LKID(KTH,1),LKID(KTH,2),LKLNG(KTH),LKLANS(KTH),
     1  LKOPOS(KTH),LKPROB(KTH,1),LKPROB(KTH,2),(LKDEST(KTH,K),K=1,3)
  101 FORMAT(A1,2I3,I4,I1,I3,2F3.3,3I3)
C
C     ..... IF IT IS A BLANK CARD, GO TO 2000.
      IF(ALPHA .EQ. 1H )  GO TO 2000
C
C     ..... FIND NODE NUMBER OF THE HEAD NODE.
C
      DO 1010 NTH=1,NUMNOS
      IF(NOIDNO(NTH) .EQ. LKID(KTH,2))  GO TO 1015
 1010 CONTINUE
      CALL DUMP(0,'INPLKS','DO1010',1)
 1015 LKNOD(KTH)=NTH
C
C     ..... SET UP THE TURNING PROBABILITIES.
C
 1060 LKPROB(KTH,2)=LKPROB(KTH,1)+LKPROB(KTH,2)
      IF(LKPROB(KTH,2) .GE. 0.9985) LKPROB(KTH,2) = 1.0
      GO TO 1000
C
C     ..... PRELIMINARY CALCULATIONS.
C
```

```
 2000 NUMLKS=KTH-1
      IF(NUMLKS .GE. MAXLKS)  CALL DUMP(NUMLKS,'INPLKS',MAXLKS,1)
C
C     ..... FIND LINK OPPOSING A LEFT TURN.
C
      DO 2500 KTH=1,NUMLKS
      IF(LKOPOS(KTH) .EQ. 0) GO TO 2500
      DO 2400 KTH2=1,NUMLKS
      IF(LKID(KTH2,2) .NE. LKID(KTH,2))  GO TO 2400
      IF(LKOPOS(KTH) .EQ. LKID(KTH2,1))  GO TO 2450
 2400 CONTINUE
      CALL DUMP(KTH,'INPLKS','LKOPOS',1)
 2450 LKOPOS(KTH)=KTH2
 2500 CONTINUE
      RETURN
      END
      SUBROUTINE INTERN(NTH)
C
C     ..... THIS SUBROUTINE READS DATA CARDS ON INTERIOR
C           NODES ONLY.  CONTROL IS RETURNED TO THE CALLING
C           PROGRAM AFTER A BLANK CARD IS FOUND.
C
      INTEGER ALPHA,TEMP
      INCLUDE DEFINS
      ITH = 0
 1000 NTH = NTH + 1
      READ(5,101) ALPHA,NOIDNO(NTH),NOTYPE(NTH),(NONXT(NTH,I),I=1,4)
      IF(ALPHA .EQ. 1H )   GO TO 3000
C
C     ..... IF NONSIGNALIZED RETURN TO READING NEXT NODE.
C
      IF(NOTYPE(NTH) .LE. 4)  GO TO 1000
C
C     ..... SIGNALIZED INTERSECTIONS REQUIRE ADDITIONAL CARD.
C
      ITH = ITH + 1
      NOSIG(NTH) = ITH
      READ(5,102) ALPHA,KDUM,SIGCYC(ITH),SIGOFF(ITH,1),SIGGRN(ITH,1),
     1  SIGOFF(ITH,2),SIGGRN(ITH,2)
C
C     ..... CALCULATE STATE CHANGE TIMES.
C
      DO 2100 I=1,2
      SIGRED(ITH,I) = SIGCYC(ITH) - SIGGRN(ITH,I) - AMBERT
      TEMP = SIGOFF(ITH,I)  + SIGGRN(ITH,I)
      IF(TEMP .GT. SIGCYC(ITH))  GO TO 2010
      TEMP = TEMP + AMBERT
      IF(TEMP .GT. SIGCYC(ITH))  GO TO 2005
      SIGSTA(ITH,I) = 2
      SIGCLK(ITH,I) = SIGOFF(ITH,I)
      GO TO 2100
 2005 SIGSTA(ITH,I) = 1
      SIGCLK(ITH,I) = TEMP - SIGCYC(ITH)
      GO TO 2100
 2010 SIGSTA(ITH,I) = 0
      SIGCLK(ITH,I) = TEMP - SIGCYC(ITH)
      GO TO 2100
```

```
 2100 CONTINUE
      GO TO 1000
 3000 NUMNOS = NTH - 1
      IF(NUMNOS .GE. MAXNO)  CALL DUMP(NUMNOS,'INTERN','MAXNO ',1)
      NUMSIG = ITH
      IF(NUMSIG .GE. MAXSIG)  CALL DUMP(NUMSIG,'INTERN','MAXSIG',1)
      RETURN
C
C     ..... FORMAT STATEMENTS
C
  101 FORMAT(A1,I3,I2,4I3,I1)
  102 FORMAT(A1,I3,5I3)
      END
      SUBROUTINE LKMOVE
C
C     ..... THIS SUBROUTINE MOVES VEHICLES ON EACH LINK.
C
      INCLUDE DEFINS
C
      DIMENSION IARM(4,2),LKEND(3,2)
      DEFINE LSTOP(KTH,LANE) = LKSTOP(KTH,LANE)
      DATA IARM/4,1,2,3,2,3,4,1/
      DATA LKEND / 40,30,12,85,60,12 /
      INTEGER VTP,VT
C**** WRITE(6,1)
    1 FORMAT(' LKMOVE')
      DO 1500 NTH=1,NUMNOS
      DO 1450 ITHARM = 1,4
      KTH = NOINP(NTH,ITHARM)
      IF(KTH .EQ. 0)  GO TO 1500
      ISW = 1
      ZBUELT = 100.
      LENGTH = LKLNG(KTH)
C
C     ..... AN EXTERNAL NODE COULD NOT BE BLOCKED
C
      IF(NOTYPE(NTH) .LT. 3)  GO TO 200
      ZB1 = 100.
      ZB2 = 100.
C
C     ..... CHECK LEFT AND RIGHT APPROACHES.
C
      DO 150 I=1,2
      INDEX = IARM(ITHARM,I)
      KTHCHK = NOINP(NTH,INDEX)
      ISW = LKLANS(KTHCHK)
      JTHCHK = LKLEAD(KTHCHK,1)
      IF(JTHCHK .EQ. 0)  GO TO 100
      IF(VPOS(JTHCHK) .GT. LKLNG(KTHCHK) + 1 )  GO TO 155
  100 IF(ISW .EQ. 1)  GO TO 150
C
C     ..... CHECK NEXT LANE.
C
      JTHCHK = LKLEAD(KTHCHK,2)
      IF(JTHCHK .EQ. 0)  GO TO 150
      IF(VPOS(JTHCHK) .GT. LKLNG(KTHCHK) + 1 )  GO TO 155
  150 CONTINUE
```

```
      GO TO 160
C
C     ..... A BLOCKAGE EXISTS
C
  155 ZB1 = 1.0
C
C     ..... NOW CHECK LINK OPPOSED
C
  160 KTHCHK = LKOPOS(KTH)
      JTHCHK = LKLEAD(KTHCHK,1)
      IF(JTHCHK .EQ. 0)  GO TO 170
      IF(VTURN(JTHCHK) .NE. 1)  GO TO 170
      IF(VPOS(JTHCHK) .LE. LKLNG(KTHCHK) + 1 + ISW*6)  GO TO 170
      ZB2 = 1.0
  170 ZBDELT = MIN(ZB1,ZB2)
  200 LANES = LKLANS(KTH)
      DO 1400 LANE = 1,LANES
      LKT = LKTAG(KTH,LANE)
C
C     ..... FIND LEADER.
C
      JTH = LKLEAD(KTH,LANE)
      IF(JTH .EQ. 0)  GO TO 1400
      JTHP = JTH
  300 X = VPOS(JTH)
      V = VASP(JTH)
      VD = VDSP(JTH)
      VT = VTURN(JTH)
      ZA = 100.
      ZB = 100.
      ZD = 100.
      ZS = 100.
      ZT = 100.
      IF(JTH .NE. JTHP)  GO TO 330
  310 IF(ZBDELT .GE. 100.)  GO TO 400
      ZB = ZDECEL(DUM,ZBDELT + LENGTH - X,V)
      GO TO 400
  330 IF(VTP .NE. 1)  GO TO 340
      IF(XP .GT. LENGTH + 20 + 6*ISW)  GO TO 310
  340 ZS = ZSPACE(X,V,XP,VP)
  400 ZA = ZACCEL(V,VD)
C
C     ..... IS LINK TAGGED.
C
      IF(LKT .EQ. 0)  GO TO 430
C
C     ..... CAN THIS VEHICLE STOP WITHIN DECELERATION LIMITS.
C
      DUM2 = 0.0
      DUM1 = ZDECEL(DUM2,LENGTH-X-12.0,V)
      IF(DUM2 .GT. 12 .OR. DUM1 .LE. 0)  GO TO 450
      LKT = 0
      LKTAG(KTH,LANE) = 0
      VTAG(JTH) = 1
      ZD = DUM1
      GO TO 500
C
```

```
C      ..... IS VEHICLE TAGGED
C
  430 IF(VTAG(JTH) .EQ. 0)  GO TO 450
C
C      ..... HAS SIGNAL CHANGED BACK TO GREEN.
C
      ITH = NOSIG(NTH)
      II = 2 - MOD(ITHARM,2)
      IF(SIGSTA(ITH,II) .NE. 0)  GO TO 435
      VTAG(JTH) = 0
      GO TO 450
  435 ZD = ZDECEL(DUM2,LENGTH-X-12.0,V)
      GO TO 500
C
C      ..... TURNING RESTRICTION
C
  450 IF(VT .EQ. 2)  GO TO 500
C**** WRITE(6,7)
    7 FORMAT('   TURNING RESTRICTION.')
      IF(X .LT. LENGTH-175)  GO TO 500
      TP = LENGTH + 1
      IF(VT .EQ. 1)  TP = TP + 6*ISW
      ZT = ZTURN (X,V,TP,MZT,VTRNEW)
C
C      ..... MOVE LIMIT.
C
  500 Z = MIN(ZA,ZB,ZD,ZS,ZT)
C**** WRITE(6,5) ZA,ZB,ZD,ZS,ZT
    5 FORMAT('   ZA,ZB,ZD,ZS,ZT: ',5F6.1)
      IF(Z .LT. 0)  Z = 0
      XNEW = X + Z
C**** WRITE(6,10) VT,NOTYPE(NTH)
   10 FORMAT('   VT AND NOTYPE ',2I4)
      IF(NOTYPE(NTH) .LT. 3)  GO TO 974
      IF(VT-2)  600,700,800
C
C      ..... LEFT TURN VEHICLES.
C
  600 TP = LENGTH + 1 + 6 * ISW
C**** WRITE(6,9) XNEW,TP,VACC(JTH)
    9 FORMAT('   LEFT TURN XNEW,TP,VACC(JTH): ',2F6.1,I3)
      IF(XNEW .LT. TP)  GO TO 1000
      IF(VACC(JTH) .NE. 0)  GO TO 610
      IF(JTHP .NE. JTH .AND. VTP .EQ. 1)  GO TO 900
C
C      ..... TEST GAP AND HOLDING AREA.
C
      CALL GAPCHK(LKOPOS(KTH),KOUT,NOSIG(NTH),2-MOD(ITHARM,2))
C**** WRITE(6,8) KOUT
    8 FORMAT('   GAPCHK OUT = ',I2)
      IF(KOUT .EQ. 1)  GO TO 900
      IF(IBHOLD(LKDEST(KTH,1)) .EQ. 0)  GO TO 900
C
C      ..... START PROCESSING THROUGH LEFT TURN.
C
      VACC(JTH) = 4
      IF(MZT .EQ. 2)  GO TO 970
```

```
      GO TO 620
C
C     .....PAST TURNING POINT AND ACCELERATING.
C
  610 IF(VACC(JTH) .NE. 1)  VACC(JTH) = VACC(JTH) - 1
  620 VNEW = V + 5 + (VACC(JTH) - 1)
      IF(VNEW .GT. VD) VNEW = VD
      ZA = 0.5 * (V + VNEW)
      XNEW = X + ZA
      DIST = XNEW - LENGTH - LKEND(1,ISW)
      IF(DIST .LT. 0)  GO TO 1000
      VACC(JTH) = 0
      KTHCHK = LKDEST(KTH,1)
      INDEX = IBHOLD(KTHCHK)
      GO TO 950
C
C     ..... STRAIGHT TRAFFIC.
C
  700 IF(XNEW .LT. LENGTH)  GO TO 1000
      KTHCHK = LKDEST(KTH,2)
      IF(IBHOLD(KTHCHK) .NE. 0)  GO TO 730
      JTHCHK = LKHOLD(KTHCHK,1)
      DUM1 = VASP(JTHCHK) - VHOLD(JTHCHK) + LENGTH + LKEND(2,ISW)
      ZS = ZSPACE(X,V,DUM1,VASP(JTHCHK))
      XNEW = MIN(XNEW,X + ZS)
      IF(XNEW .LT. X)  XNEW = X
      GO TO 1000
  730 DIST = XNEW - LENGTH - LKEND(2,ISW)
      IF(DIST .LT. 0)  GO TO 1000
      INDEX = IBHOLD(KTHCHK)
      GO TO 950
C
C     ..... RIGHT TURN VEHICLES.
C
  800 TP = LENGTH + 1
      IF(XNEW .LE. TP)  GO TO 1000
      IF(IBHOLD(LKDEST(KTH,3)) .EQ. 0)  GO TO 900
      IF(MZT .EQ. 2)  GO TO 970
      DIST = XNEW - LENGTH - LKEND(3,ISW)
      IF(DIST .LT. 0)  GO TO 1000
      KTHCHK = LKDEST(KTH,3)
      INDEX = IBHOLD(KTHCHK)
      GO TO 950
C
C     ..... STOP VEHICLE.
C
  900 XNEW = TP
      GO TO 1000
C
C     ..... VEHICLE IS GOING TO A NEW LINK.
C
  950 JTHP = VFOLOW(JTH)
      CALL HOLDIN (JTH,KTH,KTHCHK,INDEX,XNEW-X-DIST)
      IF(JTHP .EQ. 0)  GO TO 1400
      JTH = JTHP
      GO TO 300
C
```

```
C      ..... LEFT TURNING AND FLAG FROM ZTURN IS ON.
C
  970 IF(Z .NE. ZT)  GO TO 1000
      VASP(JTH) = VTRNEW
      VPOS(JTH) = XNEW
      GO TO 1010
C
C      ..... APPROACHING EXTERNAL NODE.
C
  974 IF(XNEW .LT. LENGTH)  GO TO 1000
      CALL TERMIN(JTH,KTH,LANE,JTHP)
      JTH = JTHP
      IF(JTH .EQ. 0)  GO TO 1400
      GO TO 300
C
C      ..... MOVE DESIRED DISTANCE.
C
 1000 VPOS(JTH) = XNEW
      VNEW = 2.0*(XNEW-X)-V
      IF(VNEW .GE. 4.0 .OR. VASP(JTH) .LE. 4.0)  GO TO 1005
      LSTOP(VLK(JTH),VLANE(JTH)) = LSTOP(VLK(JTH),VLANE(JTH)) + 1
 1005 VASP(JTH) = VNEW
      IF(VASP(JTH) .LT. 0) VASP(JTH) = 0.0
C**** WRITE(6,2) JTH,KTH,LANE,VPOS(JTH),VASP(JTH),VT
    2 FORMAT('   VEH ',I4,' ON LINK ',I3,I2,' POS ',F5.0,' ASP 'F5.1,
     1  ' VTURN ',I2)
 1010 JTHP = JTH
C**** WRITE(6,2) JTH,KTH,LANE,VPOS(JTH),VASP(JTH),VT
      JTH = VFOLOW(JTHP)
      IF(JTH .EQ. 0)  GO TO 1400
      VP = VASP(JTHP)
      XP = VPOS(JTHP)
      VTP = VT
      GO TO 300
C
C      ..... END OF LANE CHECK
C
 1400 CONTINUE
C
C      ..... END OF ARM CHECK
C
 1450 CONTINUE
C
C      ..... END OF NODE CHECK.
C
 1500 CONTINUE
      RETURN
      END
      FUNCTION RN1(ISEED)
C
C      ..... THIS FUNCTION GENERATES UNIFORMLY DISTRIBUTED
C            RANDOM NUMBERS BETWEEN 0 AND 1
C
      ISEED=ISEED*185333
      IF(ISEED .LT. 0) ISEED=ISEED+34359738367+1
      RN1=ISEED*2.0**(-35)
      RETURN
```

```
      END
      SUBROUTINE SETUP
C
C     ..... THIS SUBROUTINE DOES THE FINAL SETUP ON ALL DATA
C           FILES BEFORE THE SIMULATION BEGINS.
C
      INCLUDE DEFINS
C
C     ..... FOR EACH NODE
C
      DO 2100 NTH=1,NUMNOS
      IF(NOTYPE(NTH) .GT. 2) GO TO 1500
      IF(NOTYPE(NTH) .EQ. 2) GO TO 2100
C
C     ..... GENERATE NODES, SET TIME OF FIRST ARRIVAL.
C
      NOPARS(NTH,1) = 3600. / NOPARS(NTH,1)
      DO 1001 I=1,100
 1001 A = RN1(ISEED)
      NOSEED(NTH) = ISEED
      NOCLK(NTH) = TBAGEN(NTH)
C**** WRITE(6,1) NTH,NOCLK(NTH)
    1 FORMAT(' SETUP: NTH =',I2,'  NOCLK =',F7.2)
C
C     ..... FIND LINK TO PUT VEHICLES ON.
C
      DO 1100 KTH=1,NUMLKS
      IF(NONXT(NTH,1) .EQ. LKID(KTH,2) .AND. NOIDNO(NTH) .EQ. LKID(KTH,1
     1))    GO TO 1110
 1100 CONTINUE
      CALL DUMP(NTH,'SETUP ','DO1100',1)
 1110 NOOUTP(NTH) = KTH
C
C     ..... IF GENERATE ONLY SKIP TO NEXT NODE.
C
      IF(NOTYPE(NTH) .EQ. 1) GO TO 2100
C
C     ..... FIND INDEX TO LINK POINTING AT THIS NODE.
C
 1300 CONTINUE
      DO 1350 KTH=1,NUMLKS
      IF(NONXT(NTH,1) .EQ. LKID(KTH,1) .AND. NOIDNO(NTH) .EQ. LKID(KTH,2
     1))    GO TO 1360
 1350 CONTINUE
      CALL DUMP(NTH,'SETUP ','DO1350',1)
 1360 NOINP(NTH,1) = KTH
      GO TO 2100
C
C     ..... INTERNAL NODES
C
 1500 CONTINUE
      ITH = NOSIG(NTH)
      IDNTH = NOIDNO(NTH)
C
C     ..... TAKE EACH LINK AND SEE IF IT IS CONNECTED TO THIS
C           NODE.
C
```

```
      DO 1600 KTH=1,NUMLKS
C
C     ..... CHECK FOR INPUT TO THIS NODE.
C
      IF(LKID(KTH,2) .NE. IDNTH) GO TO 1600
C
C     ..... OK, NOW FIND OUT WHERE IT CAME FROM.
C
      DO 1520 I=1,4
      IF(LKID(KTH,1) .EQ. NONXT(NTH,I))  GO TO 1525
 1520 CONTINUE
      CALL DUMP(NTH,'SETUP ','DO1520',1)
 1525 NOINP(NTH,I) = KTH
      IF(ITH .NE. 0)  SIGINP(ITH,I) = KTH
 1600 CONTINUE
C
C     ..... SETUP SIGNAL LINKS.
C
      ITH = NOSIG(NTH)
      DO 2000 I=1,4
      SIGINP(ITH,I) = NOINP(NTH,I)
 2000 CONTINUE
 2100 CONTINUE
C
C     ..... FOR EACH LINK FIND THE DESTINATIONS AND SET FLAGS.
C
      DO 3000 KTH=1,NUMLKS
      LKTAG(KTH,1) = 1
      LKTAG(KTH,2) = 1
      DO 2500 I=1,3
      IF(LKDEST(KTH,I) .EQ. 0)  GO TO 2500
      DO 2400 KTHCHK=1,NUMLKS
      IF(LKID(KTHCHK,2) .NE. LKDEST(KTH,I))  GO TO 2400
      IF(LKID(KTHCHK,1) .NE. LKID(KTH,2))  GO TO 2400
      LKDEST(KTH,I) = KTHCHK
      GO TO 2500
 2400 CONTINUE
 2500 CONTINUE
 3000 CONTINUE
      RETURN
      END
      SUBROUTINE SIGCHK
C
C     ..... THIS SUBROUTINE CHECKS SIGNAL SETTINGS TO SEE IF
C           A CHANGE IN STATE IS NECESSARY.
C           POSSIBLE STATES ARE:
C                   0 - GREEN
C                   1 - AMBER
C                   2 - RED
C
      INCLUDE DEFINS
      DO 2000 ITH=1,NUMSIG
      DO 1000 I=1,2
C
C     ..... CHECK CLOCK FIRST
C
      IF(SIGCLK(ITH,I) .GT. CLOCK)  GO TO 1000
```

```
C
C     ..... BRANCH BASED ON OLD STATE.
C
      IF(SIGSTA(ITH,I)-1) 100,500,700
C
C     ..... OLD STATE WAS GREEN.
C
  100 SIGSTA(ITH,I) = 1
      SIGCLK(ITH,I) = SIGCLK(ITH,I) + AMBERT
C
C     ..... TAG INPUTS TO THIS SIGNAL.
C
      KTH = SIGINP(ITH,I)
      LKTAG(KTH,1) = 1
      IF(LKLANS(KTH) .GT. 1) LKTAG(KTH,2) = 1
      KTH = SIGINP(ITH,I+2)
      LKTAG(KTH,1) = 1
      IF(LKLANS(KTH) .GT. 1) LKTAG(KTH,2) = 1
C
C     ..... SKIP OUT TO NEXT SIGNAL.
C
      GO TO 2000
C
C     ..... OLD STATE WAS AMBER.
C
  500 SIGSTA(ITH,I) = 2
      SIGCLK(ITH,I) = SIGCLK(ITH,I) + SIGRED(ITH,I)
      GO TO 1000
C
C     ..... OLD STATE WAS RED.
C
  700 SIGSTA(ITH,I) = 0
      SIGCLK(ITH,I) = SIGCLK(ITH,I) + SIGGRN(ITH,I)
C
C     ..... NOW CHECK LENGTH OF QUEUES FOR INPUTS WHICH TURNED.
C
      KTH = SIGINP(ITH,I)
C
C     ..... CHECK FIRST LANE.
C
      LKTAG(KTH,1) = 0
      JTH = LKLAST(KTH,1)
      IF(JTH .EQ. 0)  GO TO 750
  710 IF(VASP(JTH) .LE. 4.0)  GO TO 740
      JTH = VLEAD(JTH)
      IF(JTH .EQ. 0)  GO TO 750
      GO TO 710
  740 LQ = 1
  741 JTH = VLEAD(JTH)
      IF(JTH .EQ. 0)  GO TO 745
      LQ = LQ + 1
      GO TO 741
  745 IF(LQ .GT. LKMAXQ(KTH,1))  LKMAXQ(KTH,1) = LQ
  750 IF(LKLANS(KTH) .EQ. 1)  GO TO 800
C
C     ..... CHECK SECOND LANE.
C
```

```
      LKTAG(KTH,2) = 0
      JTH = LKLAST(KTH,2)
      IF(JTH .EQ. 0)  GO TO 800
  760 IF(VASP(JTH) .LE. 4.0)  GO TO 790
      JTH = VLEAD(JTH)
      IF(JTH .EQ. 0)  GO TO 800
      GO TO 760
  790 LQ = 1
  791 JTH = VLEAD(JTH)
      IF(JTH .EQ. 0)  GO TO 795
      LQ = LQ + 1
      GO TO 791
  795 IF(LQ .GT. LKMAXQ(KTH,2))  LKMAXQ(KTH,2) = LQ
C
C     ..... NOW CHECK NEXT INPUT LINK.
C
  800 KTH = SIGINP(ITH,I+2)
C
C     ..... CHECK FIRST LANE.
C
      LKTAG(KTH,1) = 0
      JTH = LKLAST(KTH,1)
      IF(JTH .EQ. 0)  GO TO 850
  810 IF(VASP(JTH) .LE. 4.0)  GO TO 840
      JTH = VLEAD(JTH)
      IF(JTH .EQ. 0)  GO TO 850
      GO TO 810
  840 LQ = 1
  841 JTH = VLEAD(JTH)
      IF(JTH .EQ. 0)  GO TO 845
      LQ = LQ + 1
      GO TO 841
  845 IF(LQ .GT. LKMAXQ(KTH,1))  LKMAXQ(KTH,1) = LQ
  850 IF(LKLANS(KTH) .EQ. 1)  GO TO 1000
C
C     ..... CHECK SECOND LANE.
C
      LKTAG(KTH,2) = 0
      JTH = LKLAST(KTH,2)
      IF(JTH .EQ. 0)  GO TO 1000
  860 IF(VASP(JTH) .LE. 4.0)  GO TO 890
      JTH = VLEAD(JTH)
      IF(JTH .EQ. 0)  GO TO 1000
      GO TO 860
  890 LQ = 1
  891 JTH = VLEAD(JTH)
      IF(JTH .EQ. 0)  GO TO 895
      LQ = LQ + 1
      GO TO 891
  895 IF(LQ .GT. LKMAXQ(KTH,2))  LKMAXQ(KTH,2) = LQ
C     ..... NEXT CHECK THE MINOR CLOCK.
C
 1000 CONTINUE
 2000 CONTINUE
C**** WRITE(6,1) CLOCK,((SIGCLK(I,I2),SIGSTA(I,I2),I2=1,2),,=1,NUMSIG)
    1 FORMAT(//,' CLOCK ',I6,'  SIGCLK,SIGSTA '//
     1  ,16(I6,I2))
```

```
      RETURN
      END
      FUNCTION SPDDIS(ISEED)
      DIMENSION ARY(4)
      DATA ARY / -1., 0.07, 0.97, 1.0 /
      X=RN1(ISEED)
      DO 100 I=2,4
      IF(ARY(I) .GE. X)  GO TO 200
  100 CONTINUE
  200 SPDDIS = I * 18.0
      RETURN
      END
      SUBROUTINE STAT(K)
C
C     ..... THIS SUBROUTINE CLEARS THE STATISTICS AFTER WARMUP
C           AND THEN PRINTS THEM AFTER THE FINISH.
C
      INCLUDE DEFINS
C
C     ..... IF FINISH, GO TO 1000
C
      IF(K .EQ. 2)  GO TO 1000
      DO 200 KTH = 1,NUMLKS
      DO 100 I=1,2
      LKMAXQ(KTH,I) = 0
      LKSTOP(KTH,I) = 0
      LKDEL(KTH,I) = 0
  100 LKVL(KTH,I) = 0
      DO 200 I=1,3
      LKDELD(KTH,I) = 0
  200 LKVD(KTH,I) = 0
      RETURN
C
C     ..... PRINT THE STATISTICS IN THIS SECTION.
C
 1000 TOTIM = FINISH - WARMUP
      WRITE(6,1)
    1 FORMAT(1H1,21X,'VEHICLE COUNT',15X,'VEHICLE VOLUME',2X,
     1  'VEHICLE DELAY',14X,'DELAY PER VEHICLE',/,' LINK   LANES   ',
     2  'LANE 1   LANE 2   TOTAL     LANE 1   LANE 2   TOTAL',
     3  12X,'LANE 1   LANE 2   TOTAL       LANE 1   LANE 2   TOTAL',/)
      DO 1100 KTH=1,NUMLKS
      LKVTOT = LKVL(KTH,1) + LKVL(KTH,2)
      LKVOL1 = LKVL(KTH,1) * 3600 / TOTIM
      LKVOL2 = LKVL(KTH,2) * 3600 / TOTIM
      LKVOLT = LKVOL1 + LKVOL2
      LKDELT = LKDEL(KTH,1) + LKDEL(KTH,2)
      DELVL1 = LKDEL(KTH,1) / FLOAT(LKVL(KTH,1))
      DELVL2 = LKDEL(KTH,2) / FLOAT(LKVL(KTH,2))
      DELVLT = LKDELT / FLOAT(LKVL(KTH,1) + LKVL(KTH,2))
 1100 WRITE(6,2) KTH,LKLANS(KTH),LKVL(KTH,1),LKVL(KTH,2),LKVTOT,
     1  LKVOL1,LKVOL2,LKVOLT,LKDEL(KTH,1),LKDEL(KTH,2),LKDELT,
     2  DELVL1,DELVL2,DELVLT
    2 FORMAT(I4,4X,I2,2(1X,3I9),9X,3I9,2X,3F9.2)
      WRITE(6,3)
    3 FORMAT(///,14X,'VEHICLE STOPS',11X,'MAX QUEUES',14X,'TURNS',
     1  20X,'TURN DELAY',14X,'TURN DELAY PER VEHICLE',/,
```

```
     2  ' LINK    LANE 1    LANE 2    TOTAL     LANE 1    LANE 2       LEFT',
     3  ' STRAIGHT  RIGHT       LEFT    STRAIGHT    RIGHT      LEFT',
     4  '  STRAIGHT    RIGHT',/)
      DO 1200 KTH=1,NUMLKS
      LKSTOT = LKSTOP(KTH,1) + LKSTOP(KTH,2)
      DELVL1 = LKDELD(KTH,1) / LKVD(KTH,1)
      DELVL2 = LKDELD(KTH,2) / LKVD(KTH,2)
      DELVL3 = LKDELD(KTH,3) / LKVD(KTH,3)
 1200 WRITE(6,4) KTH,LKSTOP(KTH,1),LKSTOP(KTH,2),LKSTOT,LKMAXQ(KTH,1),
     1  LKMAXQ(KTH,2),(LKVD(KTH,I),I=1,3),(LKDELD(KTH,I),I=1,3),
     2  DELVL1,DELVL2,DELVL3
    4 FORMAT(I4,3(I7,2X),6X,I2,7X,I2,2X,3(3X,I5),2(4X,I6),3X,I6,
     1  2X,3F9.2)
    5 FORMAT(///,23H INTERSECTION ID NUMBER,I4,/,12H0INPUT LINKS,6X,
     1  14HVEHICLE VOLUME,10X,9HMAX QUEUE,11X,13HVEHICLE DELAY,12X,
     2  17HDELAY PER VEHICLE,/,14X,22HLANE 1  LANE 2   TOTAL,4X,
     3  14HLANE 1  LANE 2,2(4X,22HLANE 1  LANE 2   TOTAL),/)
    6 FORMAT(I7,4X,2I8,I9,2I8,3X,2I8,I9,2X,2F8.2,F9.2)
    7 FORMAT(30X,6H------,64X,6H------,/,21H  TOTAL INTERSECTION ,
     1   8HVOLUME = ,I7,23X,40HAVERAGE INTERSECTION DELAY PER VEHICLE =,
     2   F7.2)
    8 FORMAT(/////,32H AVERAGE LINK DELAY FOR SYSTEM = F7.2)
    9 FORMAT(29H1INTERSECTION OUTPUT SUMMARY.)
      WRITE(6,9)
      NTOT = 0
      NDTOT = 0
      DO 1275 NTH=1,NUMNOS
      IF(NOTYPE(NTH) .LT. 3)  GO TO 1275
      NINT = 0
      NDEL = 0
      WRITE(6,5) NOIDNO(NTH)
      ITH = NOSIG(NTH)
      DO 1250 II=1,4
      KTH = SIGINP(ITH,II)
      LKVTOT = LKVL(KTH,1) + LKVL(KTH,2)
      NINT = NINT + LKVTOT
      LKDELT = LKDEL(KTH,1) + LKDEL(KTH,2)
      NDEL = NDEL + LKDELT
      DELVL1 = LKDEL(KTH,1) / FLOAT(LKVL(KTH,1))
      DELVL2 = LKDEL(KTH,2) / FLOAT(LKVL(KTH,2))
      DELVLT = LKDELT / FLOAT(LKVTOT)
 1250 WRITE(6,6) KTH,LKVL(KTH,1),LKVL(KTH,2),LKVTOT,LKMAXQ(KTH,1),
     1           LKMAXQ(KTH,2),LKDEL(KTH,1),LKDEL(KTH,2),LKDELT,
     2           DELVL1,DELVL2,DELVLT
      NTOT = NTOT + NINT
      NDTOT = NDTOT + NDEL
      DEL = NDEL / FLOAT(NINT)
      WRITE(6,7) NINT,DEL
 1275 CONTINUE
      DEL = NDTOT / FLOAT(NTOT)
      WRITE(6,8) DEL
      RETURN
      END
      SUBROUTINE STATA(I)
C
C     ..... PERIOD BY PERIOD OUTPUT GENERATOR.  LKSOUT IS AN ARRAY
C           TELLING THE DESIRED OUTPUT LINKS.
```

```
C
      INCLUDE DEFINS
      DIMENSION LKSOUT(4),N1(4),AVG1(4),X1(4),L1(4),LL1(4),LS1(4),LR1(4)
     1   ,DEL1(4),NL1(4),NS1(4),NR1(4),DELY1(4)
      DATA LKSOUT / 1,5,0,0 /
      PARAMETER N=2
      GO TO (10,20),I
   10 CONTINUE
C**** WRITE(6,4)
      DO 15 II=1,N
      L1(II)=0
      DEL1(II)=0
      LL1(II)=0
      LS1(II)=0
   15 LR1(II)=0
      RETURN
C**** WRITE(6,1) CLOCK,NUMVEH
   20 CONTINUE
      DO 30 II=1,N
      MM = LKSOUT(II)
      L1P = LKVL(MM,1) + LKVL(MM,2)
      N1(II)=L1P - L1(II)
      L1(II)=L1P
      NL1(II)=LKVD(MM,1)-LL1(II)
      LL1(II)=LKVD(MM,1)
      NS1(II)=LKVD(MM,2)-LS1(II)
      LS1(II)=LKVD(MM,2)
      NR1(II)=LKVD(MM,3)-LR1(II)
      LR1(II)=LKVD(MM,3)
      IDELP = LKDEL(MM,1) + LKDEL(MM,2)
      DELY1(II)=IDELP-DEL1(II)
      DEL1(II)=IDELP
      AVG1(II)=DELY1(II)/N1(II)
      X1(II)=IDELP/FLOAT(L1P)
C**** WRITE(6,2) L1P,(LKVD(MM,J),J=1,3),
C****1  IDELP,X1(II),N1(II),NL1(II),NS1(II),NR1(II),AVG1(II)
   30 CONTINUE
      PUNCH 3,CLOCK,NUMVEH,((N1(II),AVG1(II),X1(II)),II=1,N)
      RETURN
    1 FORMAT(/,9H CLOCK = ,I7,13H    NUMVEH = ,I7)
    2 FORMAT(10X,I6,3I5,I8,F10.2,I10,3I5,F8.2)
    3 FORMAT(I7,I5,4(I5,2F6.1))
    4 FORMAT(1H1,12X,10HCUMULATIVE,18X,7HCURRENT,7X,11HLAST PERIOD//,
     1  13X,25HVOL   LF   ST   RT  DELAY,7X,3HMOE,7X,
     2  25HVOL   LF   ST   RT  DEL/V)
      END
      FUNCTION TBAGEN(NTH)
C
C     ..... THIS FUNCTION GENERATES THE TIME BETWEEN ARRIVALS
C           AT THE NTH NODE.  AN EXPONENTIAL TIME BETWEEN
C           ARRIVALS IS ASSUMED.
C
      INCLUDE DEFINS
      TBAGEN = -ALOG(RN1(NOSEED(NTH))) * NOPARS(NTH,1)
      RETURN
      END
      SUBROUTINE TERMIN(JTH,KTH,LANE,JTHCHK)
```

```
C
C     ..... THIS SUBROUTINE TERMINATES VEHICLES THAT ARE
C           LEAVING THE NETWORK.
C
      INCLUDE DEFINS
C
C     ..... FIRST WORK ON LINK.
      JTHCHK = VFOLOW(JTH)
      LKLEAD(KTH,LANE) = JTHCHK
      IF(JTHCHK .NE. 0)  VLEAD(JTHCHK) = 0
      IF(JTHCHK .EQ. 0)  LKLAST(KTH,LANE) = 0
C
C     ..... NEXT ZERO DESIRED SPEED AND LINK NUMBER.
      VDSP(JTH) = 0
      VLK(JTH) = 0
      VTURN(JTH) = 0
      VASP(JTH) = 0
      VFOLOW(JTH) = 0
      NUMVEH = NUMVEH - 1
C**** WRITE(6,1) JTH,KTH,JTHCHK
    1 FORMAT('   TERMINATE VEHICLE ',I4,' FROM LINK',I3,'VNEXT ',I3)
C
C     ..... COULD GATHER SOME STATISTICS IF DESIRED.
C
      RETURN
      END
      SUBROUTINE VEGEN
C
C     ..... THIS SUBROUTINE CHECKS THE CLOCK AT EACH GENERATE
C           NODE TO SEE IF IT IS TIME TO GENERATE ANOTHER
C           VEHICLE.
C
      INCLUDE DEFINS
C**** WRITE(6,1)
    1 FORMAT(' VEGEN SUBROUTINE.')
      DO 2000 NTH=1,NUMNOS
      IF(NOTYPE(NTH) .GT. 1)  GO TO 2000
C
C     ..... CHECK TO SEE IF READY FOR ANOTHER VEHICLE
C
 1000 IF (CLOCK .LT. NOCLK(NTH))  GO TO 2000
C
C     ..... FIND OUT IF HOLDING AREA IS AVAILABLE.
      KTH = NOOUTP(NTH)
      INDEX = IRHOLD(KTH)
      IF(INDEX .EQ. 0)  GO TO 2000
C
C     ..... HOLDING AREA READY, FIND AN AVAILABLE VEHICLE.
C
 1105 CONTINUE
      DO 1110 JTH=1,MAXVEH
      IF(VDSP(JTH) .LE. 0)  GO TO 1121
 1110 CONTINUE
      CALL DUMP(0,'VEGEN ','MAXVEH',27)
C
C     ..... NOW FILL IN HIS PARAMETERS.
C
```

```
 1121 VDSP(JTH) = SPDDIS(ISFED)
      VASP(JTH) = VDSP(JTH)
      NUMVEH = NUMVEH + 1
C
C     ..... PLACE VEHICLE IN HOLDING AREA
C
C**** WRITE(6,2) JTH,NTH,VDSP(JTH)
    2 FORMAT('   VEH ',I4,'  AT NODE ',I2,'  VDSP ',F4.0)
      CALL HOLDIN(JTH,0,KTH,INDEX,0.0)
C
C     .....COMPUTE TIME OF NEXT ARRIVAL
C
      NOCLK(NTH) = NOCLK(NTH) + TBAGEN(NTH)
C
C     ..... READY FOR ANOTHER.
C
      GO TO 1000
 2000 CONTINUE
      RETURN
      END
      FUNCTION ZACCEL(V,VMAX)
C
C     ..... ACCELERATION RESTRICTION CALCULATION.
C
      PARAMETER ABAR = 4
      VNEW = V + ABAR
      IF(VNEW .GT. VMAX)  VNEW = VMAX
      ZACCEL = 0.5*(V+VNEW)
      RETURN
      END
      FUNCTION ZDECEL(DTEST,XGAP,V)
C
C     ..... STOPPING RESTRICTION.
C
      PARAMETER DBAR = 6
      IF(XGAP .LE. 0)  GO TO 15
      DTEST = V**2 / (2.0*XGAP)
      IF(DTEST .GE. DBAR)  GO TO 30
   15 ZDECEL = XGAP
      RETURN
   30 R = 0.0625*DTEST**2 - 0.25*DTEST*V + 0.5*DTEST*XGAP
      IF(R .GT. 0)  GO TO 35
      R = 0
      GO TO 40
   35 R = SQRT(R)
   40 ZDECEL = 0.5*V - 0.25*DTEST + R
      RETURN
      END
      FUNCTION ZSPACE(X,V,XP,VP)
C
C     ..... SPACING RESTRICTION CALCULATIONS.
C
      PARAMETER DBAR = 6
      PARAMETER PMIN = 22
C
C     ..... BRANCH ON CASE TYPE.
C
```

```
      IF(V .LE. VP) GO TO 30
C
C     ..... CASE I.
C
   10 R =  9*DBAR**2/16 - DBAR*V/4.0 - .75*DBAR*VP + DBAR*(VP-X-PMIN)/2.
      IF(R .GT. 0)  GO TO 15
      R = 0
      GO TO 20
   15 R = SQRT(R)
   20 ZSPACE = .5*(V+VP) - .75*DBAR + R
      RETURN
C
C     ..... CASE II.
C
   30 ZSPACE = (XP-X-PMIN+V)/3.0
      RETURN
      END
      FUNCTION ZTURN(X,V,TP,MZT,VTRNEW)
C
C     ..... THIS FUNCTION COMPUTES THE DESIRED MOVE THAT
C           WOULD BE REQUIRED TO SLOW TO THE TURNING SPEED
C           BY THE TURNING POINT.
C
      PARAMETER DBAR = 6
      PARAMETER VMAX = 12
      PARAMETER ABAR = 3
      MZT = 1
      B = TP - X
      IF(B .LE. 0)  GO TO 40
      R = DBAR**2/16. + VMAX**2 *0.25 - DBAR*V*0.25 + 0.5*DBAR*B
            IF(R .LE. 0)  GO TO 20
      R = SQRT(R)
      GO TO 25
   20 R = 0
   25 ZTURN = 0.5 * V - 0.25*DBAR + R
      IF(ZTURN .LE. B)  GO TO 50
C
C     ..... VEHICLE PASSED THE TURNING POINT.
C
      VMXTP = SQRT(V**2 + 2*ABAR*B)
      IF(VMXTP - 15.0) 40,40,30
C
C     ..... COMPUTE REVISED ZTURN AND SPECIAL V.
C
   30 BT = 1.0 -(B+R)/(V+15.0)
      ZTURN = B + 15.0*BT + 1.5*BT**2
      VTRNEW = 15.0 + 3.0*BT
      MZT = 2
      RETURN
   40 ZTURN = 100.0
   50 RETURN
      END
```

APPENDIX C

FORTRAN LISTING OF UB MODEL

```
DEFINS PROCEDURE
C
C     ..... SETUP FOR NODES.
C
      PARAMETER MAXNO=35
      REAL NOPARS,NOCLK
      COMMON /NODES/ NOTDNO(MAXNO),NOTYPE(MAXNO),NONXT(MAXNO,4),
     1  NOPARS(MAXNO,2),NOGFOM(MAXNO),NOINP(MAXNO,4),NOOUTP(MAXNO),
     2  NOSEED(MAXNO),NOFSTB(MAXNO),NOSIG(MAXNO),NOCLK(MAXNO),NUMNOS
C
C     ..... SETUP FOR SIGNALS.
C
      PARAMETER MAXSIG=20
      INTEGER AMBERT,SIGCLK,SIGCYC,SIGGRN,SIGINP,SIGOFF,SIGRED,SIGSTA
      COMMON /SIG/ SIGCLK(MAXSIG,2),SIGCYC(MAXSIG),SIGINP(MAXSIG,4),
     1  SIGGRN(MAXSIG,2),SIGOFF(MAXSIG,2),SIGRED(MAXSIG,2),
     2  SIGSTA(MAXSIG,2),NUMSIG,AMBERT
C
C     ..... SETUP FOR LINKS.
C
      PARAMETER MAXLKS=100
      REAL LKPROB
      COMMON /LINKS/ LKTD(MAXLKS,2),LKLNG(MAXLKS),LKLANS(MAXLKS),
     1  LKFSTB(MAXLKS),LKNOD(MAXLKS),LKOPOS(MAXLKS),
     2  LKPROB(MAXLKS,2),LKDEST(MAXLKS,3),LKVD(MAXLKS,3),LKVL(MAXLKS,2),
     3  LKDEL(MAXLKS,2),LKDELD(MAXLKS,3),LKMAXQ(MAXLKS,2),
     4  LKSTOP(MAXLKS,2),NUMLKS
C
C     ..... SETUP FOR BLOCKS.
C
      PARAMETER MAXBLK=5000
      INTEGER BLKARY
      COMMON /BLOCKS/ BLKARY(MAXBLK),LENGTH,NBLKS
C
C     ..... SETUP FOR VEHICLES.
C
      PARAMETER MAXVEH=2000
      INTEGER VDSP,VASP,VMOVE,VSTATE,VTURN,VLK,VHOLD,VLAN,VLKTIM
      COMMON/VEH/VDSP(MAXVEH),VASP(MAXVEH),VMOVE(MAXVEH),VSTATE(MAXVEH),
     1  VTURN(MAXVEH),VLK(MAXVEH),VHOLD(MAXVEH),VLAN(MAXVEH),NUMVEH,
     2  VLKTIM(MAXVEH)
C
C     ..... GENERAL COMMON VARIABLES.
C
      INTEGER CLOCK,DELT,FINISH,WARMUP
      COMMON CLOCK,ISEED,DELT,FINISH,WARMUP
C
C     *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C     *                                                             *
C     *     DEFINITION OF VARIABLES USED IN THE PROGRAM.            *
C     *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C     *                                                             *
C     *                                                             *
C     *                                                             *
C     *     AMBERT - AMBER TIME.
C     *     BLKARY(IB) - ARRAY OF BLOCKS.                           *
```

```
C     *     CLOCK - MASTER SIMULATION CLOCK.                          *
C     *     DELT - THE TIME STEP.
C     *     FINISH - RUNNING TIME (IN SECONDS).
C     *     ITH - INDEX USED TO POINT TO SPECIFIC SIGNAL.             *
C     *     KTH - INDEX USED TO POINT TO SPECIFIC LINK.               *
C     *     LENGTH - LENGTH OF A UNIT BLOCK (IN FEET).                *
C     *     LKDEST(KTH,1-3) - LINK DESTINATIONS: LEFT,STR,RIGHT.      *
C     *     LKFSTB(KTH) - LINK'S FIRST BLOCK.                         *
C     *     LKID(KTH,1-2) - SYMBOLIC I.D. OF TAIL(1) AND HEAD(2).     *
C     *     LKLANS(KTH) - NUMBER OF LANES.                            *
C     *     LKLNG(KTH) - LINK LENGTH (IN BLOCKS).                     *
C     *     LKNOD(KTH) - INDEX TO NODE AT HEAD OF LINK.               *
C     *     LKNTYP(KTH) - TYPE OF NODE APPROACHING:                   *
C     *                       0 - NO REASON TO STOP IF STRAIGHT,      *
C     *                       1 - APPROACHING A SIGNAL,               *
C     *                       2 - APPROACHING A STOP SIGN.            *
C     *     LKOPOS(KTH) - LINK APPOSING A LEFT TURN.                  *
C     *     LKPROB(KTH,1-2) - CUMULATIVE PROB OF LEFT & STR. MOVE.    *
C     *     MAXBLK - MAXIMUM POSSIBLE NUMBER OF BLOCKS.               *
C     *     MAXLKS - MAXIMUM POSSIBLE NUMBER OF LINKS.                *
C     *     MAXNO  - MAXIMUM POSSIBLE NUMBER OF NODES.                *
C     *     MAXSIG - MAXIMUM POSSIBLE NUMBER OF SIGNALS.              *
C     *     MAXVEH - MAXIMUM POSSIBLE NUMBER OF VEHICLES.             *
C     *     NBLKS - NUMBER OF BLOCKS IN MODEL.                        *
C     *     NOCLK(NTH) - TIME OF NEXT VEGEN.
C     *     NOFSTB(NTH) - NODES FIRST BLOCK.                          *
C     *     NOGEOM(NTH) - GEOMETRY OF NODE:                           *
C     *                       EXTERNAL = NUMBER OF LANES,             *
C     *                       INTERNAL = TYPE OF INTERSECTION:        *
C     *                             1 - TWO BY TWO,                   *
C     *                             2 - FOUR BY TWO,                  *
C     *                             3 - FOUR BY FOUR.                 *
C     *     NOIDNO(NTH) - SYMBOLIC NODE IDENTIFIER.                   *
C     *     NOINP(NTH,1-4) - INDEX TO LINKS INPUTTING TO THIS NODE    *
C     *     NONXT(NTH,1-4) - SYMBOLIC IDENTIFIERS OF ADJ. NODES.      *
C     *     NOOUTP(NTH) - INDEX TO LINKS TAKING FROM THIS NODE        *
C     *     NOPARS(NTH,1-2) - PARAMETERS FOR GENERATE NODES.          *
C     *     NOSIG(NTH) - INDEX TO SIGNAL FOR NTH NODE.                *
C     *     NOTYPE(NTH) - NODE TYPE:                                  *
C     *                       0 - BOTH GENERATE AND TERMINATE,        *
C     *                       1 - GENERATE ONLY,                      *
C     *                       2 - TERMINATE ONLY,                     *
C     *                       5 - FIXED TIME CONTROLLED.              *
C     *     NTH - INDEX USED TO POINT TO SPECIFIC NODE.               *
C     *     NUMLKS - NUMBER OF LINKS IN MODEL.                        *
C     *     NUMNOS - NUMBER OF NODES.                                 *
C     *     NUMSIG - NUMBER OF SIGNALS.                               *
C     *     NUMVEH - NUMBER OF VEHICLES IN NETWORK.                   *
C     *     SIGCLK(ITH,1-2) - TIME OF NEXT SIGNAL CHANGE.
C     *     SIGCYC(ITH) - CYCLE LENGTH OF ITH SIGNAL.
C     *     SIGGRN(ITH,1-2) - GREEN TIME.
C     *     SIGINP(ITH,1-4) - LINKS POINTING AT ITH SIGNAL.
C     *     SIGOFF(ITH,1-2) - SIGNAL OFFSET.
C     *     SIGRED(ITH,1-2) - RED TIME.
C     *     SIGSTA(ITH,1-2) - SIGNAL STATE:
C     *               0 - GREEN
C     *               1 - AMBER
```

```
C     *                2 - RED.
C     *     VASP(JTH) - VEHICLE'S ACTUAL SPEED.                        *
C     *     VDSP(JTH) - VEHICLE'S DESIRED SPEED.                       *
C     *     VHOLD(JTH) - NUMBER OF BLOCKS MOVED BEFORE HOLDING         *
C     *     VLAN(JTH) - LANE DESIRED.                                  *
C     *     VLK(JTH) - INDEX OF CURRENT OR MOST RECENT LINK.           *
C     *     VLKTIM(JTH) - CLOCK TIME VEHICLE SHOULD LEAVE LINK.
C     *     VMOVE(ITH) - MOVE FLAG, 0 - NOT PROCESSED THIS PERIOD .
C     *     VSTATE(JTH) - STATE OF JTH VEHICLE:                        *
C     *                          0 - STOPPED, MUST HESITATE,           *
C     *                          1 - ACCELERATING,                     *
C     *                          2 - DECELERATING                      *
C     *                          3 - CONSTANT SPEED.                   *
C     *     VTURN(JTH) - TURN INDICATOR:                               *
C     *                          1 - LEFT,                             *
C     *                          2 - STRAIGHT,                         *
C     *                          3 - RIGHT.                            *
C     *     WARMUP - TIME TO LOAD THE NETWORK.
C     *                                                                *
C     *                                                                *
C     *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C     *                                                                *
C     *                                                                *
C     *                                                                *
C     *                                                                *
C     *                                                                *
C     *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C
C
C
 END
C
C     ..... THIS IS THE MAIN PROGRAM FOR THE UNIT BLOCK MODEL.
C           CLOCK IN SECONDS.
C
C
      INCLUDE DEFINS,LIST
C
C     ..... INITIALIZATION
      NTH = 0
      DELT = 1
      CLOCK = -DELT
      NBLKS = 0
      LENGTH = 18
C
C     ..... RUN PARAMETERS
C
      READ(5,1) IRUN,INET,AMBERT,ISEED,FINISH,WARMUP
    1 FORMAT( )
      MODEL = '  UB'
      WRITE(21) MODEL,IRUN,INET,ISEED
      WRITE(22) MODEL,IRUN,INET,ISEED
      PUNCH 6,IRUN,INET,ISEED
    6 FORMAT('UB MODEL    IRUN = ',I4,'  INET = 'I4,'  ISEED = ',I12)
      WARMUP = WARMUP * 60
      FINISH = FINISH * 60 + WARMUP
      WRITE(6,2) IRUN,INET,ISEED,FINISH,WARMUP
    2 FORMAT('1**************** UNIT BLOCK MODEL **********',//
```

```
     1  ' RUN NUMBER ',I5,//,'   NETWORK ',I5,'   ISEED ',I9,
     2  //,'   FINISH = ',I8,' SECONDS.  WARMUP =',I4,' SECONDS.',//)
C
C      ..... CALL INPUT SUBROUTINES.
C
       CALL EXTERN(NTH)
       CALL INTERN(NTH)
       CALL INPLKS
C
C      ..... CALL SETUP TO FINISH DATA MANIPULATION.
C
       CALL SETUP
       CALL DUMP(0,'DATA C','HECK  ',0)
C
C      ..... THE SIMULATION.
C
 1000  CLOCK = CLOCK + DELT
       IF(MOD(CLOCK,90).EQ.0)  CALL STATA(2)
       WRITE(22) CLOCK,NUMVEH
       IF(CLOCK  .GE.  WARMUP) GO TO 2000
       CALL VCLEAR
       CALL SIGCHK
       CALL VEGEN
       CALL INTERL
       CALL INTRAL
       CALL HOLDOU
       GO TO 1000
C
C      ..... WARMUP OVER, CLEAR STATISTICS.
C
 2000  CALL STAT(1)
       CALL STATA(1)
       GO TO 2200
 2100  CLOCK = CLOCK + DELT
       IF(MOD(CLOCK,90).EQ.0)  CALL STATA(2)
       WRITE(22) CLOCK,NUMVEH
       IF(CLOCK  .GE.  FINISH) GO TO 3000
 2200  CALL VCLEAR
       CALL SIGCHK
       CALL VEGEN
       CALL INTERL
       CALL INTRAL
       CALL HOLDOU
       GO TO 2100
C
C      ..... SIMULATION OVER, PRINT STATISTICS.
C
 3000  CALL STAT(2)
       ENDFILE 21
       ENDFILE 22
       STOP
       END
       SUBROUTINE ADVCHK(JTH,JTHLDR,NUMPOS,NUM)
C
C      ..... THIS SUBROUTINE CALCULATES HOW FAR VEHICLES CAN
C            ADVANCE KNOWING SIZE OF GAP AND VEHICLES INVOLVED.
C
```

```
      INCLUDE DEFINS
      DIMENSION IGAPA(4,5)
      DATA IGAPA/0,1,3,6,1,2,4,6,0,2,4,4,0,0,2,4,0,0,0,3/
      DEFINE IGAP(I,J)=IGAPA(I,J+1)
      NUMDS = VASP(JTH)
      IF(VASP(JTH) .LT. VDSP(JTH) .AND. VSTATE(JTH) .NE. 1)  NUMDS =
     1          NUMDS + 1
      DO 1000 I=NUMDS,1,-1
      NUM = I
      IF(NUMPOS-NUM .GE. IGAP(NUM,VASP(JTHLNR)))  RETURN
 1000 CONTINUE
      NUM = 0
 1010 RETURN
      END
      SUBROUTINE DISTCH(JTH,KTH,POS,NTH,NUM2,ITH,IARM)
C
C     ..... THIS SUBROUTINE CHECKS THE JTH VEHICLE WHICH IS
C           IN POSITION POS OF THE KTH LINK APPROACHING THE
C           NTH NODE.  NUM2 IS THE NUMBER OF POSITIONS HE
C           SHOULD MOVE CONSIDERING ONLY THE INTERSECTION.
C
      INCLUDE DEFINS
C
      DIMENSION ISQ2(4,2,4)
      INTEGER POS,EFFECT(4),TURNSP(10),ISQ(4,3),STOPSP(10)
      DATA ISQ /3,4,2,1,4,2,1,3,2,1,3,4 /
      DATA ISQ2 /9,10,11,12,13,14,15,16, 15,11,7,3,16,12,8,4,
     1           8,7,6,5,4,3,2,1,  2,6,10,14,1,5,9,13/
      DATA EFFECT /4,6,8,10 /
      DATA TURNSP /1,2,2,3,3,4,4,4,4,4 /
      DATA STOPSP /0,1,1,2,2,3,3,3,4,4/
      II = 2 - MOD(IARM,2)
C
C     ..... DETERMINE FASTEST SPEED DESIRED.
      NUM2 = VASP(JTH)
      IF(NUM2 .LT. VDSP(JTH) .AND. VSTATE(JTH) .NE. 1)  NUM2 = NUM2 + 1
C
C     ..... CHECK THE POSITION OF THE VEHICLE.
      IF(POS .GT. EFFECT(NUM2))  RETURN
      IF(NOTYPE(NTH) .LE. 2)  RETURN
      IF(NOGEOM(NTH) .EQ. 3)  GO TO 3000
C
C
C     .....TWO-BY-TWO INTERSECTION.
C
 1050 IF(SIGSTA(ITH,II) .NE. 0)  GO TO 9900
      IF(VTURN(JTH) .EQ. 2)  GO TO 1065
 1055 IF(NUM2 .GT. TURNSP(POS))  NUM2 = TURNSP(POS)
 1065 IF(POS-NUM2 .GT. 0)  RETURN
      NUMIN = NUM2 - POS + 1
      IF(VTURN(JTH)-2)  2200,2100,2000
C
C     ..... RIGHT TURNS.
C
 2000 IB = ISQ(IARM,1) + NOFSTB(NTH) - 1
      IF(BLKARY(IB) .EQ. 0)  GO TO 2010
      GO TO 9800
```

```
 2010 IF(IBHOLD(LKDEST(KTH,3)) .EQ. 0)  GO TO 9800
      RETURN
C
C     ..... STRAIGHT MOVEMENTS.
C
 2100 IB = ISQ(IARM,2) + NOFSTB(NTH) - 1
      IF(BLKARY(IB) .EQ. 0)  GO TO 2110
      JTHLDR = BLKARY(IB)
      IF(VLK(JTHLDR) .NE. KTH)  GO TO 9800
      IF(VTURN(JTHLDR) .EQ. 2)  GO TO 2106
      GO TO 9800
 2106 CALL ADVCHK(JTH,JTHLDR,POS,NUM1)
      IF(NUM1 .LT. NUM2) NUM2 = NUM1
 2110 IB = ISQ(IARM,1) + NOFSTB(NTH) - 1
      IF(BLKARY(IB) .NE. 0)  GO TO 9800
      IF(NUMIN .LE. 2)  RETURN
      IF(IBHOLD(LKDEST(KTH,2)) .NE. 0)   RETURN
      KTHLDR = LKDEST(KTH,2)
      IB = LKFSTB(KTHLDR) + LKLNG(KTHLDR)
      JTHLDR = BLKARY(IB)
      CALL ADVCHK(JTH,JTHLDR,POS+1,NUM2)
      RETURN
C
C     ..... LEFT TURNS.
C
 2200 IF(IBHOLD(LKDEST(KTH,1)) .EQ.0)  GO TO 9800
      IB = ISQ(IARM,2) + NOFSTB(NTH) - 1
      IF(BLKARY(IB) .EQ. 0)  GO TO 2210
      JTHLDR = BLKARY(IB)
      IF(VLK(JTHLDR) .NE. KTH)  GO TO 9800
      IF(VTURN(JTHLDR) .EQ. 1)  GO TO 9800
 2210 IB = ISQ(IARM,1) + NOFSTB(NTH) - 1
      IF(BLKARY(IB) .NE. 0)  GO TO 9800
      RETURN
C
C     ..... FOUR BY FOUR INTERSECTIONS.
C
 3000 IF(SIGSTA(ITH,II) .NE. 0)  GO TO 9900
      IF(VTURN(JTH) .EQ. 2)  GO TO 3100
      IF(NUM2 .GT. TURNSP(POS))  NUM2 = TURNSP(POS)
 3100 IF(POS-NUM2 .GT. 0)  RETURN
C
C     ..... VEHICLE GOING INTO THE INTERSECTION.
C
      IF(VLAN(JTH) .EQ. 1)  GO TO 3500
C
C     ..... RIGHT LANE, MAY TURN RIGHT OR GO STRAIGHT.
C
      IF(VTURN(JTH) .EQ. 2)  GO TO 3200
C
C     ..... RIGHT TURNS.
C
      IB = ISQ2(1,2,IARM) + NOFSTB(NTH) - 1
      IF(BLKARY(IB) .NE. 0)  GO TO 9800
      IF(IBHOLD(LKDEST(KTH,3)) .EQ. 0)  GO TO 9800
      RETURN
C
```

```
C     ..... STRAIGHT TRAFFIC IN THE RIGHT LANE.
C
 3200 NGAP = POS - 1
      DO 3210 I=1,4
      IB = ISQ2(I,2,IARM) + NOFSTB(NTH) - 1
      IF(BLKARY(IB) .NE. 0)  GO TO 3220
 3210 NGAP = NGAP + 1
      RETURN
 3220 JTHCHK = BLKARY(IB)
      IF(VLK(JTHCHK) .NE. KTH)  GO TO 9800
      CALL ADVCHK(JTH,JTHCHK,NGAP,NUM1)
      IF(NUM1 .LT. NUM2)  NUM2 = NUM1
      RETURN
C
C     LEFT LANE, MAY TURN LEFT OR GO STRAIGHT.
C
 3500 ISTOP = VTURN(JTH) + 2
      NGAP = POS - 1
      DO 3510 I=1,ISTOP
      IB = ISQ2(I,1,IARM) + NOFSTB(NTH) - 1
      IF(BLKARY(IB) .NE. 0)  GO TO 3520
 3510 NGAP = NGAP + 1
      RETURN
 3520 JTHCHK = BLKARY(IB)
      IF(VLK(JTHCHK) .NE. KTH) GO TO 9800
      IF(VTURN(JTHCHK) .EQ. 1)  GO TO 9800
      CALL ADVCHK(JTH,JTHCHK,NGAP,NUM1)
      IF(NUM1 .LT. NUM2)  NUM2 = NUM1
      RETURN
 9800 NUM2 = POS - 1
      RETURN
 9900 IF(NUM2 .GT.STOPSP(POS))  NUM2 = STOPSP(POS)
      RETURN
      END
      SUBROUTINE DUMP(M,N1,N2,K)
C
C     ..... THIS SUBROUTINE PROVIDES A DUMP OF THE VARIABLES
C           IN CASE OF AN ERROR.  THE VALUE OF K DETERMINES
C           THE TYPE OF DUMP:
C                              0 - SET UP CHECK, CALL RETURN
C                                  AFTER PRINTOUT ON NODES, ETC.
C                              1 - ERROR ON INPUT,
C                              2 - ERROR WHILE RUNNING.
C
      INCLUDE DEFINS
C
C     ..... GENERAL INFORMATION DUMP.
      WRITE(6,1) K
      IF(K .GE. 2)  GO TO 1000
      WRITE(6,2) M,N1,N2
      GO TO 1001
 1000 WRITE(6,3) M,N1,N2
      L = MAXNO
 1001 WRITE(6,4) L,NUMNOS
      DO 1010 I=1,NUMNOS
 1010 WRITE(6,5) NOIDNO(I),NOTYPE(I),(NONXT(I,K),K=1,4),NOPARS(I,1),
     1NOPARS(I,2),NOGEOM(I),(NOINP(I,K),K=1,4),NOOUTP(I),
```

```
     2NOFSTB(I),NOSIG(I),NOCLK(I)
      L = MAXLKS
      WRITE(6,6) L,NUMLKS
      DO 1020 I=1,NUMLKS
 1020 WRITE(6,7) LKID(I,1),LKID(I,2),LKLNG(I),LKLANS(I),LKFSTB(I),
     1LKNOD(I),LKOPOS(I),LKPROB(I,1),LKPROB(I,2),LKDEST(I,1),
     2LKDEST(I,2),LKDEST(I,3)
      WRITE(6,8) NBLKS,(BLKARY(I),I=1,NBLKS)
      IF(K .EQ. 0) RETURN
      IF(K .EQ. 1) STOP DUMP1
      WRITE(6,9) CLOCK
      DO 1090 J=1,MAXVEH
      IF(VDSP(J) .EQ. 0)  GO TO 1090
      IDSP=VDSP(J)
      IASP=VASP(J)
      IMOVE=VMOVE(J)
      ISTATE=VSTATE(J)
      ITURN=VTURN(J)
      ILK=VLK(J)
      IHOLD=VHOLD(J)
      ILAN=VLAN(J)
      WRITE(6,10) J,IDSP,IASP,IMOVE,ISTATE,ITURN,ILK,IHOLD,ILAN
 1090 CONTINUE
      STOP DUMP2
1     FORMAT(///' ERROR DUMP OUTPUT.  K =',I5)
2     FORMAT(' CODE ',I5,'  MESSAGE - ',2A6)
3     FORMAT(' VEHICLE',I6,', NOW BEING PROCESSED.',/,' MESSAGE - ',2A6)
4     FORMAT(///,' NODE OUTPUT, MAX =',I4,' NUMBER =',I5,//,2X,'ID',2X,
     1'TYP',3X,'NEXT NODE',2X,'PARAMETERS  GEO',3X,'LINKS IN',4X,
     2'OUT',2X,'FSTB',2X,'SIG  CLK',/)
5     FORMAT(2X,I2,3X,I1,3X,4(I2,1X),2(1X,F4.0),3X,I1,3X,4(I2,1X),1X,I
     12,3X,I3,3X,I2,3X,F4.0)
6     FORMAT(///,' LINK OUTPUT.  MAX =',I4,' NUMBER =',I5,//,3X,'ID',3X,
     1'LENGTH  LANES  FSTB  NODE',3X,'OPOS   PROBABILITIES',3X,
     2'DESTINATIONS',/)
7     FORMAT(2X,I2,1X,I2,2X,I3,6X,I1,4X,I4,3X,I2,4X,I3,2X,2(2X,F5.3),
     1   4X,3(I3,2X))
8     FORMAT(///,' BLOCK OUTPUT.',/,(1X,25I5))
9     FORMAT(///,' CLOCK =',I7,///,' VEHICLES IN NETWORK.',/,'  VEH   DS
     1P  ASP MOVE  STA TURN  LK  HOLD  LAN',/)
10    FORMAT(10I5)
      END
      SUBROUTINE EXTERN(NTH)
C
C     ..... THIS SUBROUTINE READS THE DATA CARDS ON EXTERNAL
C           NODES ONLY.  CONTROL IS RETURNED TO THE CALLING
C           PROGRAM AFTER A BLANK CARD IS FOUND
C
      INCLUDE DEFINS
      INTEGER ALPHA
 1000 NTH=NTH+1
      READ(5,101) ALPHA,NOIDNO(NTH),NOTYPE(NTH),NONXT(NTH,1),
     1  NOPARS(NTH,1),NOPARS(NTH,2),NOGEOM(NTH)
      IF(ALPHA .EQ. 1H )  GO TO 2000
      GO TO 1000
 2000 NTH=NTH-1
      RETURN
```

```
  101 FORMAT(A1,I3,I2,I3,2F5.0,I1)
      END
      SUBROUTINE GAPCHK(KTH,KOUT,NTH,ITH,II)
C
C     ..... THIS SUBROUTINE FINDS THE SIZE OF THE GAP ON THE
C           KTH LINK.
C
      INCLUDE DEFINS
      DIMENSION PROB(20)
      DATA PROB / -1., -1., .02, .10, .22, .36, .50, .61, .74,
     1    .81, .84, .90, .92, .95, .96, .97, .98, .99, .995, .9995  /
      KOUT = 0
      IF(NOTYPE(NTH) .LE. 2)  RETURN
      IF(SIGSTA(ITH,II) .NE. 0)  RETURN
 1200 IF(LKLANS(KTH) .EQ. 2)  GO TO 1300
C
C     ..... SINGLE LANE STREET.
C
      IBSTRT = LKFSTB(KTH)
      IBSTOP = IBSTRT + LKLNG(KTH) - 1
      DO 1250 IB=IBSTRT,IBSTOP
      IF(BLKARY(IB) .NE. 0)  GO TO 1251
 1250 CONTINUE
      RETURN
 1251 JTH = BLKARY(IB)
      IF(VASP(JTH) .EQ. 0)  RETURN
 1260 GAP  = (IB - IBSTRT) / FLOAT(VASP(JTH))
      GO TO 1600
C
C     ..... TWO LANE APPROACH.
C
C     ..... CHECK THE FIRST LANE.
C
 1300 GAP1 = 999.
      IBSTRT = LKFSTB(KTH)
      IBSTOP = IBSTRT + LKLNG(KTH) - 1
      DO 1310 IB=IBSTRT,IBSTOP
      IF(BLKARY(IB) .NE. 0)  GO TO 1311
 1310 CONTINUE
      GO TO 1400
 1311 JTH = BLKARY(IB)
      IF(VASP(JTH) .EQ. 0)  GO TO 1400
 1320 GAP1 = (IB - IBSTRT) / FLOAT(VASP(JTH))
C
C     ..... CHECK THE SECOND LANE.
C
 1400 GAP2 = 999.
      IBSTRT = IBSTOP + 1
      IBSTOP = IBSTRT + LKLNG(KTH) - 1
      DO 1420 IB=IBSTRT,IBSTOP
      IF(BLKARY(IB) .NE. 0)  GO TO 1421
 1420 CONTINUE
      GO TO 1500
 1421 JTH = BLKARY(IB)
      IF(VASP(JTH) .EQ. 0)  GO TO 1500
 1430 GAP2 = (IB - IBSTRT) / FLOAT(VASP(JTH))
 1500 GAP = MIN(GAP1,GAP2)
```

```
C
C     ..... CHECK THE SIZE OF THE GAP.
C
 1600 IF(GAP .GE. 20.0)  RETURN
      IF(GAP .LE. 2)  GO TO 2000
      INDEX = GAP  + 0.5
      IF(RN1(ISEED) .LE. PROB(INDEX))  RETURN
C
C     ..... REJECT.
 2000 KOUT = 1
      RETURN
      END
      SUBROUTINE GETSTA(JTH,KTHOLD,KTH,ITURN)
C
C     ..... THIS SUBROUTINE CALCULATES THE STATISTICS FOR THE
C           LINK THAT THE VEHICLE HAS JUST EXITED.  THE MIN TIME
C           DEPENDS ON THE TURNING SPEED RESTRICTION IN THE DISTCH
C           SUBROUTINE.
C
      INCLUDE DEFINS
      DIMENSION ICON(4)
      DATA ICON / 0,2,6,9 /
      DEFINE ICONP(I) = ICON(I)
      IF(KTHOLD .EQ. 0)  GO TO 5
C
C     ..... GET STATISTICS ON OLD LINK.
C
      LANE = VLAN(JTH)
      IDELAY = CLOCK - VLKTIM(JTH)
      WRITE(21) CLOCK,KTHOLD,JTH,IDELAY
C**** WRITE(6,2) JTH,VLKTIM(JTH),CLOCK,IDELAY
    2 FORMAT(' #####  IDELAY OUTPUT ',6I6)
      LKVL(KTHOLD,LANE) = LKVL(KTHOLD,LANE) + 1
      LKVD(KTHOLD,ITURN) = LKVD(KTHOLD,ITURN) + 1
      LKDEL(KTHOLD,LANE) = LKDEL(KTHOLD,LANE) + IDELAY
      LKDELD(KTHOLD,ITURN) = LKDELD(KTHOLD,ITURN) + IDELAY
C
C     ..... CALCULATE TIME EXPECTED TO TRAVEL LINK AND
C           TIME VEHICLE SHOULD REACH END OF LINK.
C
    5 IDSP = VASP(JTH)
      IF(IDSP .LT. VDSP(JTH))  IDSP = IDSP + 1
      NUM = IDSP - VHOLD(JTH) - 1
C
C     ..... BRANCH BASED ON TURNING MOVEMENT.
C
      ITURN = VTURN(JTH)
      GO TO (10,20,30),ITURN
C
C     ..... LEFT TURN CALCULATIONS DEPEND AND NUMBER OF LANES.
C
   10 MINTIM = (LKLNG(KTH) - NUM + ICONP(VDSP(JTH))) / VDSP(JTH)
     1                      + LKLANS(KTH) * 2
      GO TO 40
C
C     ..... STRAIGHT MOVEMENTS.
C
```

```
   20 MINTIM = (LKLNG(KTH) - NUM + LKLANS(KTH) * 2 + VDSP(JTH) - 1) /
     1                          VDSP(JTH)
      GO TO 40
C
C     ..... RIGHT TURNS ARE NOT EFFECTED BY NUMBER OF LANES.
C
   30 MINTIM = (LKLNG(KTH) - NUM + ICONP(VDSP(JTH))) / VDSP(JTH)
   40 VLKTIM(JTH) = CLOCK + MINTIM
    1 FORMAT(' $$$$$$ GETSTA OUTPUT',7I6)
C**** WRITE(6,1) JTH,KTHOLD,KTH,ITURN,LANE,IDELAY,VLKTIM(JTH)
      RETURN
      END
      SUBROUTINE HOLDIN(JTH,IBTO,IBFROM,I)
C
C     ..... THIS SUBROUTINE MOVES A VEHICLE INTO A HOLDING AREA.
C           IT WILL BE REMOVED AND THE STATISTICS UPDATED BY
C           THE HOLDOUT SUBROUTINE AFTER ALL LINK MOVEMENT
C           HAS TAKEN PLACE.
C
      INCLUDE DEFINS
C
      BLKARY(IBTO) = JTH
      IF(IBFROM .NE. 0)   BLKARY(IBFROM) = 0
C
C     ..... VHOLD IS USED TO STORE THE NUMBER OF BLOCKS ALREADY MOVED.
      VHOLD(JTH) = I
C**** WRITE(6,1) JTH,IBTO,IBFROM,I
    1 FORMAT('   HOLDIN VEH ',I4,' TO BLK ',I4,' FROM BLK',I5,
     1  ' VHOLD ',I2)
      RETURN
      END
      SUBROUTINE HOLDOU
C
C     ..... THIS SUBROUTINE TAKES VEHICLES OUT OF A HOLDING
C           AREA AND MOVES THEM ONTO THE LINK.
C
      INCLUDE DEFINS
      DO 3000 KTH=1,NUMLKS
      LANE = 1
C
C     ..... BRANCH ON NUMBER OF LANES
      IF(LKLANS(KTH) .EQ. 2)  GO TO 2000
C
C     ..... SINGLE LANE.  LOOK AT HOLDING AREA. IF EMPTY SKIP.
      IB = LKFSTB(KTH) + LKLNG(KTH)
      IF(BLKARY(IB) .EQ. 0)  GO TO 3000
C
C     ..... IF NEW VEHICLE TO THIS LINK, ASSIGN PARAMETERS.
      JTH = BLKARY(IB)
      IF(VLK(JTH) .EQ. KTH)  GO TO 1100
      KTHOLD = VLK(JTH)
      ITURN = VTURN(JTH)
      VLK(JTH) = KTH
      VLAN(JTH) = 1
      X = RN1(ISEED)
      DO 1050 I=1,2
      IF(X .LE. LKPROB(KTH,I))  GO TO 1055
```

```
 1050 CONTINUE
      VTURN(JTH) = 3
      GO TO 1075
 1055 VTURN(JTH) = 1
 1075 CALL GETSTA(JTH,KTHOLD,KTH,ITURN)
C
C     ..... FIND LEADING VEHICLE.
 1100 IBSTOP = LKFSTB(KTH)
      IBSTRT = IB - 1
      DO 1110 IBCHK=IBSTRT,IBSTOP,-1
      IF(BLKARY(IBCHK) .NE. 0)  GO TO 1200
 1110 CONTINUE
C
C     ..... NO LEADER ADVANCE DESIRED NUMBER OF BLOCKS
      NUM = VASP(JTH)
      IF(NUM .LT. VASP(JTH) .AND. VSTATE(JTH) .NE. 1)  NUM = NUM + 1
      IBNEW = IB - NUM + VHOLD(JTH)
      IF(IBNEW .LT. LKFSTB(KTH))  CALL DUMP(JTH,'HOLDOU','NOLEAD',40)
      IF(IBNEW .EQ. IB)  GO TO 2975
      IF(BLKARY(IBNEW) .NE. 0)  CALL DUMP(JTH,'HOLDOU','FULBLK',42)
      GO TO 2950
C
C     ..... LEADER FOUND.
C
 1200 NUMPOS = IB - IBCHK - 1 + VHOLD(JTH)
      IF(NUMPOS .LE. 0)  GO TO 2975
      JTHLDR = BLKARY(IBCHK)
      CALL ADVCHK(JTH,JTHLDR,NUMPOS,NUM)
      NUMPOS = NUM - VHOLD(JTH)
      IF(NUMPOS .LE. 0)  GO TO 2975
      IBNEW = IB - NUMPOS
      GO TO 2950
C
C
C     ..... TWO LANE LINK PROCESSING. LOOK AT HOLDING AREAS.
C
 2000 IB = LKFSTB(KTH) + LKLNG(KTH) * 2
      IF(BLKARY(IB) .NE. 0)  GO TO 2100
 2050 IB = IB + 1
      LANE = LANE + 1
      IF(BLKARY(IB) .NE. 0)  GO TO 2100
      IF(LANE .GE. LKLANS(KTH))  GO TO 3000
      GO TO 2050
C
C     ..... FOUND A VEHICLE IN THE HOLDING AREA.  IF VEHICLE
C           NEW TO THIS LINK, ASSIGN PARAMETERS.
C
 2100 JTH = BLKARY(IB)
      IF(VLK(JTH) .EQ. KTH)  GO TO 2150
      KTHOLD = VLK(JTH)
      ITURN = VTURN(JTH)
      VLK(JTH) = KTH
      X = RN1(ISEED)
      IF(X .GT. LKPROB(KTH,1))  GO TO 2140
      VTURN(JTH) = 1
      VLAN(JTH) = 1
      GO TO 2149
```

```
 2140 IF(X .GT. LKPROB(KTH,2))  GO TO 2145
      IF(VTURN(JTH) .EQ. 2)  GO TO 2149
      VTURN(JTH) = 2
      VLAN(JTH) = RN1(ISEED) * LKLANS(KTH) + 1
      GO TO 2149
 2145 VTURN(JTH) = 3
      VLAN(JTH) = LKLANS(KTH)
 2149 CALL GETSTA(JTH,KTHOLD,KTH,ITURN)
C
C     ..... FIND LEADER IN DESIRED LANE.
C
 2150 IBSTOP = LKFSTB(KTH) + LKLNG(KTH) * (VLAN(JTH) - 1)
      IBSTRT = IBSTOP + LKLNG(KTH) - 1
      DO 2160 IBCHK=IBSTRT,IBSTOP,-1
      IF(BLKARY(IBCHK) .NE. 0)  GO TO 2175
 2160 CONTINUE
C
C     ..... NO LEADER FOUND ADVANCE
      NUM = VASP(JTH)
      IF(NUM .LT. VDSP(JTH) .AND. VSTATE(JTH) .NE. 1)  NUM = NUM + 1
      IBNEW = IBSTRT + 1 - NUM + VHOLD(JTH)
      IF(IBNEW .LT. IBSTOP)  IBNEW = IBSTOP
      IF(IBNEW .GT. IBSTRT)  GO TO 2975
      IF(BLKARY(IBNEW) .NE. 0) CALL DUMP(JTH,'HOLDOU','FULBLK',107)
      GO TO 2950
C
C     ..... LEADER FOUND
 2175 NUMPOS = IBSTRT - IBCHK + VHOLD(JTH)
      IF(NUMPOS .LE. 0)  GO TO 2975
      JTHLDR = BLKARY(IBCHK)
      CALL ADVCHK(JTH,JTHLDR,NUMPOS,NUM)
      NUMPOS = NUM - VHOLD(JTH)
      IF(NUMPOS .LE. 0)  GO TO 2975
      IBNEW = IBSTRT + 1 - NUMPOS
      GO TO 2950
C
C     ..... MOVE VEHICLE FORWARD NUM BLOCKS
C
 2950 CONTINUE
      VHOLD(JTH) = 0
      BLKARY(IBNEW) = JTH
      BLKARY(IB) = 0
      CALL UPDATV(JTH,NUM,1)
C**** WRITE(6,1) JTH,IB,IBNEW,NUM
      IF(LANE .LT. LKLANS(KTH))  GO TO 2050
      GO TO 3000
C
C     ..... VEHICLE CAN NOT MOVE OUT OF HOLDING AREA.
C
 2975 CONTINUE
      VHOLD(JTH) = 0
      CALL UPDATV(JTH,0,1)
      NZERO = 0
C**** WRITE(6,1) JTH,IB,IB,NZERO
1     FORMAT(' HOLDOU',4I6)
      IF(LANE .LT. LKLANS(KTH))  GO TO 2050
 3000 CONTINUE
```

```
      RETURN
      END
      FUNCTION IBHOLD(KTH)
C
C     ..... THIS FUNCTION SEARCHES THE HOLDING AREA OF THE
C           KTH LINK.  IF THERE IS AN EMPTY SPACE IN THE
C           HOLDING AREA, THE BLOCK ADDRESS IS RETURNED AS
C           THE VALUE OF THE FUNCTION.  IF THE AREA IS FULL,
C           ZERO IS THE RETURNED VALUE OF THE FUNCTION.
C
      INCLUDE DEFINS
C
C     ..... BRANCH BASED ON THE NUMBER OF LANES.
      IBHOLD = 0
      IF(LKLANS(KTH) .EQ. 2)  GO TO 1000
C
C     ..... SINGLE LANE, IF HOLDING AREA FULL THEN RETURN.
      IB = LKFSTB(KTH) + LKLNG(KTH)
      IF(BLKARY(IB) .NE. 0)  RETURN
      IBHOLD = IB
      RETURN
C
C     ..... TWO LANES - MUST CHECK BOTH HOLDING BLOCKS.
 1000 IB = LKFSTB(KTH) + LKLNG(KTH) * 2
      IF(BLKARY(IB) .NE. 0)  GO TO 1010
      IBHOLD = IB
      RETURN
 1010 IB = IB + 1
      IF(BLKARY(IB) .NE. 0)  RETURN
      IBHOLD = IB
      RETURN
      END
      SUBROUTINE INPLKS
C
C     ..... THIS SUBROUTINE READS CARDS FOR ALL NETWORK LINKS.
C           AFTER ALL DATA CARDS HAVE BEEN READ, THE ROUTINE
C           CALCULATES SOME CROSS REFERENCE INDICES.
C
      INCLUDE DEFINS
C
C     ..... THIS IS THE LOCAL DECLARATION.
C
      INTEGER ALPHA
      KTH=0
 1000 KTH=KTH+1
      READ(5,101) ALPHA,LKID(KTH,1),LKID(KTH,2),LKLNG(KTH),LKLANS(KTH),
     1  LKOPOS(KTH),LKPROB(KTH,1),LKPROB(KTH,2),(LKDEST(KTH,K),K=1,3)
C
C     ..... IF IT IS A BLANK CARD, GO TO 2000.
      IF(ALPHA .EQ. 1H )  GO TO 2000
C
C     ..... CALCULATE THE BLOCKS PER LANE AND THE NUMBER OF
C           BLOCKS USED THUS FAR.
      LKLNG(KTH)=LKLNG(KTH)/LENGTH
      LKFSTB(KTH)=NBLKS+1
      NBLKS=NBLKS+LKLNG(KTH)*LKLANS(KTH)+LKLANS(KTH)
C
```

```
C     ..... FIND NODE NUMBER OF THE HEAD NODE.
C
      DO 1010 NTH=1,NUMNOS
      IF(NOIDNO(NTH) .EQ. LKID(KTH,2))  GO TO 1015
 1010 CONTINUE
      CALL DUMP(KTH,'INPLKS','D01010',1)
 1015 LKNOD(KTH)=NTH
C
C     ..... SET UP THE TURNING PROBABILITIES.
C
 1060 LKPROB(KTH,2)=LKPROB(KTH,1)+LKPROB(KTH,2)
      IF(LKPROB(KTH,2) .GE. 0.9985) LKPROB(KTH,2) = 1.0
      GO TO 1000
C
C     ..... FIX THE NUMBER OF LINKS IN NETWORK.
C
 2000 NUMLKS=KTH-1
      IF(NUMLKS .GE. MAXLKS)  CALL DUMP(NUMLKS,'INPLKS','MAXLKS',1)
      IF(NBLKS .GT. MAXBLK)  CALL DUMP(NBLKS,'INPLKS','MAXBLK',1)
C
C     ..... FIND THE LINK OPPOSING A LEFT TURN.
C
      DO 2500 KTH=1,NUMLKS
      IF(LKOPOS(KTH) .EQ. 0) GO TO 2500
      DO 2400 KTH2=1,NUMLKS
      IF(LKID(KTH2,2) .NE. LKID(KTH,2))  GO TO 2400
      IF(LKOPOS(KTH) .EQ. LKID(KTH2,1))  GO TO 2450
 2400 CONTINUE
      CALL DUMP(KTH,'INPLKS','LKOPOS',1)
 2450 LKOPOS(KTH)=KTH2
 2500 CONTINUE
      RETURN
  101 FORMAT(A1,2I3,I4,I1,I3,2F3.3,3I3)
      END
      SUBROUTINE INTERL
C
C     ..... THIS SUBROUTINE CALLS THE HANDLER APPROPRIATE TO
C           THE NODE TYPE.
C
      INCLUDE DEFINS
      DO 2000 NTH=1,NUMNOS
C
C     ..... IF EXTERNAL NODE FORGET IT AND GO TO THE NEXT ONE.
C
      IF(NOTYPE(NTH) .LE. 2)  GO TO 2000
C
C     ..... SET UP A COMPUTED GO TO BASED ON NODE GEOMETRY.
C
      K=NOGEOM(NTH)
      GO TO (1100,1200,1300),K
C
C     ..... TWO-BY-TWO.
 1100 CALL MOVETT(NTH)
      GO TO 2000
C
C     ..... FOUR-BY-TWO.
 1200 CALL DUMP(NTH,'INTERL',' 4X2  ',23)
```

```
      GO TO 2000
C
C     ..... FOUR-BY-FOUR.
 1300 CALL MOVEFF(NTH)
 2000 CONTINUE
      RETURN
      END
      SUBROUTINE INTERN(NTH)
C
C     ..... THIS SUBROUTINE READS THE DATA CARDS ON INTERIOR
C           NODES ONLY.  CONTROL IS RETURNED TO THE CALLING
C           PROGRAM AFTER A BLANK CARD IS FOUND.
C
      INTEGER ALPHA
      INCLUDE DEFINS
      ITH=0
 1000 NTH=NTH+1
      READ(5,101) ALPHA,NOIDNO(NTH),NOTYPE(NTH),(NONXT(NTH,K),K=1,4),
     1  NOGEOM(NTH)
C
      IF(ALPHA .EQ. 1H )  GO TO 3000
C
C     ..... PORTION OFF A SECTION OF BLOCKS FOR THE INTERSECTION.
      NOFSTB(NTH)=NBLKS+1
      K=NOGEOM(NTH)
      GO TO (1100,1200,1300),K
 1100 NBLKS=NBLKS+4
      GO TO 1900
 1200 NBLKS=NBLKS+8
      GO TO 1900
 1300 NBLKS=NBLKS+16
 1900 CONTINUE
      IF(NOTYPE(NTH) .LE. 4)  GO TO 1000
C
C     ..... SIGNALIZED INTERSECTIONS NEED TO READ ANOTHER DATA
C           CARD TO GET CONTROL INFORMATION.
      ITH=ITH+1
      NOSIG(NTH)=ITH
      READ(5,102) ALPHA,KDUM,SIGCYC(ITH),SIGOFF(ITH,1),SIGGRN(ITH,1),
     1  SIGOFF(ITH,2),SIGGRN(ITH,2)
C
C     ..... CALCULATE STATE CHANGE TIMES AND PUT ON CHAIN.
C
      DO 2100 I=1,2
      SIGRED(ITH,I) = SIGCYC(ITH) - SIGGRN(ITH,I) - AMBERT
      TEMP = SIGOFF(ITH,I)  + SIGGRN(ITH,I)
      IF(TEMP .GT. SIGCYC(ITH))  GO TO 2010
      TEMP = TEMP + AMBERT
      IF(TEMP .GT. SIGCYC(ITH))  GO TO 2005
      SIGSTA(ITH,I) = 2
      SIGCLK(ITH,I) = SIGOFF(ITH,I)
      GO TO 2100
 2005 SIGSTA(ITH,I) = 1
      SIGCLK(ITH,I) = TEMP - SIGCYC(ITH)
      GO TO 2100
 2010 SIGSTA(ITH,I) = 0
      SIGCLK(ITH,I) = TEMP - SIGCYC(ITH)
```

```
      GO TO 2100
 2100 CONTINUE
      GO TO 1000
 3000 NUMNOS = NTH - 1
      IF(NUMNOS .GE. MAXNO)  CALL DUMP(NUMNOS,'INTERN','MAXNO ',1)
      NUMSIG = ITH
      IF(NUMSIG .GE. MAXSIG)  CALL DUMP(NUMSIG,'INTERN','MAXSIG',1)
      RETURN
C
C     ..... FORMAT STATEMENTS
C
  101 FORMAT(A1,I3,I2,4I3,I1)
  102 FORMAT(A1,I3,5I3)
      END
      SUBROUTINE INTRAL
C
C     ..... THIS SUBROUTINE CHECKS EACH LINK FOR MOVEMENT
C           WITHIN THE LINK.
C
      INCLUDE DEFINS
      DIMENSION NUADD(4,2),ISQ2(4,2,4)
      DATA ISQ2 /9,10,11,12,13,14,15,16,   15,11,7,3,16,12,8,4,
     1     8,7,6,5,4,3,2,1,   2,6,10,14,1,5,9,13/
      DATA NUADD /3,4,2,1,4,2,1,3/
      INTEGER POS
C
C     ..... SEARCH EACH LINK IN NETWORK.
C
      DO 6000 KTH=1,NUMLKS
      LANE = 0
      NTH = LKNOD(KTH)
      ITH = NOSIG(NTH)
C
C     ..... FIND THE ARM THE VEHICLE IS ON.
C
      DO 560 IARM=1,4
      IF(NOINP(NTH,IARM) .EQ. KTH)  GO TO 561
  560 CONTINUE
      CALL DUMP(JTH,'INTRAL',' ARM  ',27)
  561 CONTINUE
 1000 LANE = LANE + 1
C
C     ..... SET UP THE BLOCK LIMITS FOR THIS LANE.
C
      IBSTRT = LKFSTB(KTH) + (LANE - 1) * LKLNG(KTH)
      IBSTOP = IBSTRT + LKLNG(KTH) - 1
      POS = 0
      JTHLDR = 0
C
C     ..... STEP THROUGH EACH BLOCK ON THE LINK.
C
      DO 5000 IB=IBSTRT,IBSTOP
      POS = POS + 1
C
C     ..... IF NO VEHICLE, SKIP TO THE NEXT BLOCK.
C
      IF(BLKARY(IB) .EQ. 0)  GO TO 5000
```

```
      JTH = BLKARY(IB)
      NUM1 = VASP(JTH)
      IF(NUM1 .LT. VDSP(JTH) .AND. VSTATE(JTH) .NE. 1)  NUM1=NUM1+1
      CALL DISTCH(JTH,KTH,POS,NTH,NUM2,ITH,IARM)
      NUM = MIN(NUM1,NUM2)
      IF(JTHLDR .EQ. 0)  GO TO 1525
C
C     ..... FOR FOLLOWERS ONLY.
C
 1500 NUMPOS = IB -IBLDR - 1
      CALL ADVCHK(JTH,JTHLDR,NUMPOS,NUM)
C
C     ..... BOTH LEADERS AND FOLLOWERS.
C
 1525 IF(NUM .EQ. 0)  GO TO 4990
      IF(VSTATE(JTH) .EQ. 0)  GO TO 4980
      IF(POS - NUM .GE. 1)  GO TO 4975
C
C     ..... VEHICLE CAN MOVE, BUT THE INTERSECTION IS
C           INVOLVED.  BRANCH ON INTERSECTION GEOMETRY.
C
      IF(NOTYPE(NTH) .LE. 2)  GO TO 4970
      IF(NOGEOM(NTH) .EQ. 3)  GO TO 2000
C
C     ..... A TWO-BY-TWO INTERSECTION IS INVOLVED.
C
      NUMDS = NUM - POS + 1
C
C     ..... BRANCH TO SECTION WITH PROPER DESTINATION.
C
 1561 IF(NUMDS .EQ. 1)  GO TO 1800
      IF(VTURN(JTH) .NE. 3)  GO TO 1580
      I = 3
      NUM = POS
      GO TO 1850
 1580 IF(NUMDS .EQ. 2)  GO TO 1825
      I = VTURN(JTH)
      NUM = POS + 1
      GO TO 1850
C
C     ..... MOVE FORWARD ONE SQUARE INTO THE INTERSECTION.
C
 1800 IBNEW = NOFSTB(NTH) + NUADD(IARM,1) - 1
      GO TO 4971
C
C     ..... MOVE FORWARD TWO SQUARES INTO THE INTERSECTION.
C
 1825 IBNEW = NOFSTB(NTH) + NUADD(IARM,2) - 1
      GO TO 4971
C
C     ..... ADVANCE INTO THE HOLDING AREA.
C
 1850 KTHNEW = LKDEST(KTH,I)
      IBNEW = IBHOLD(KTHNEW)
      IF(IBNEW .EQ. 0)  CALL DUMP(JTH,'INTRAL','IBHOLD',105)
      CALL HOLDIN(JTH,IBNEW,IB,NUM)
      GO TO 5000
```

```
C
C     ..... A FOUR BY FOUR INTERSECTION IS INVOLVED.
C
 2000 NUMDS = NUM - POS + 1
      IF(NUMDS .NE. 1)  GO TO 2100
C
C     ..... MOVE FORWARD ONE SQUARE INTO INTERSECTION.
C
      IBNEW = NOFSTR(NTH) - 1 + ISQ2(1,LANE,IARM)
      GO TO 4971
C
C     ..... MORE THAN ONE SQUARE.
C
 2100 IF(VTURN(JTH) .NE. 2)  CALL DUMP(JTH,'INTRAL','TURNSP',126)
C
      IF(NUMDS .GT. 4)  GO TO 2200
 2150 IBNEW = NOFSTR(NTH) - 1 + ISQ2(NUMDS,LANE,IARM)
      GO TO 4971
C
C     ..... TRY TO CROSS INTERSECTION INTO HOLDING AREA.
C
 2200 IBNEW = IBHOLD(LKDEST(KTH,2))
      IF(IBNEW .NE. 0)  GO TO 2250
      I = NUMDS - 4
      NUMDS = 4
      NUM = NUM - I
      GO TO 2150
 2250 CALL HOLDIN(JTH,IBNEW,IB,NUM)
      GO TO 5000
C
C     ..... VEHICLE IS GOING TO A TERMINATE NODE.
C
 4970 CALL TERMIN(JTH,IB)
      GO TO 5000
C
C     ..... VEHICLE IS GOING INTO THE INTERSECTION.
C
 4971 BLKARY(IB) = 0
      BLKARY(IBNEW) = JTH
      CALL UPDATV(JTH,NUM,1)
C**** WRITE(6,1) JTH,IB,IBNEW,NUM
      GO TO 5000
C
C     ..... SIMPLE MOVE OF 'NUM' BLOCKS ON SAME LINK.
C
 4975 IBNEW = IB - NUM
      BLKARY(IB) = 0
      BLKARY(IBNEW) = JTH
      CALL UPDATV(JTH,NUM,1)
C**** WRITE(6,1) JTH,IB,IBNEW,NUM
      GO TO 4995
C
C     ..... HESITATE THIS TIME, CAN MOVE NEXT TIME.
C
 4980 VSTATE(JTH) = 3
      IBNEW = IB
      CALL UPDATV(JTH,0,1)
```

```
      NZERO = 0
C**** WRITE(6,1) JTH,IB,IB,NZERO
      GO TO 4995
C
C     ..... SIT STILL AND WAIT.
C
 4990 IBNEW = IB
      CALL UPDATV(JTH,0,0)
C**** WRITE(6,1) JTH,IB,IB,NZERO
1     FORMAT(' INTRAL',4I6)
C
C     ..... MAKE THIS VEHICLE THE CURRENT LEADER.
C
 4995 JTHLDR = JTH
      IBLDR = IBNEW
C
C     ..... RETURN TO TOP AND LOOK AT NEXT BLOCK.
C
 5000 CONTINUE
C
C     ..... ANOTHER LANE.
C
      IF(LANE .LT. LKLANS(KTH))  GO TO 1000
 6000 CONTINUE
      RETURN
      END
      SUBROUTINE MOVEFF(NTH)
C
C     ..... THIS SUBROUTINE MOVES VEHICLES IN A 4X4 INTERSECTION.
C           THE RED OR AMBER VEHICLES ARE CHECKED FIRST, THEN
C           THOSE FACING A GREEN LIGHT.
C
      INCLUDE DEFINS
      INTEGER ORDER,LEFTSQ,POS,STRTSQ
      DIMENSION ORDER(4,2),LEFTSQ(5,4),STRTSQ(4,2,4)
      DATA ORDER /1,3,2,4,2,4,1,3 /
      DATA LEFTSQ /3,7,11,4,8, 5,6,7,1,2, 14,10,6,13,9, 12,11,10,16,15 /
      DATA STRTSQ / 9,10,11,12,13,14,15,16,  15,11,7,3,16,12,8,4,
     1     8,7,6,5,4,3,2,1,   2,6,10,14,1,5,9,13 /
C
C     ..... SETUP TO GET RED AND AMBER FIRST.
C
      ITH = NOSIG(NTH)
      IF(SIGSTA(ITH,1)-1)  10,20,30
   10 J = 2
      GO TO 50
   20 J = 1
      GO TO 50
   30 IF(SIGSTA(ITH,2)-2) 20,10,20
   50 IBASE = NOFSTR(NTH) - 1
C
C     ..... TAKE EACH APPROACH LINK AND LOOK FOR CARS FROM
C           THAT LINK IN THE INTERSECTION.
C
      DO 1000 I=1,4
      IARM = ORDER(I,J)
      KTH = NOINP(NTH,IARM)
```

```
C
C     ..... FIRST GET RIGHT TURNS.
C
      IB = IBASE + STRTSQ(1,2,IARM)
      IF(BLKARY(IB) .EQ. 0)  GO TO 100
      JTH = BLKARY(IB)
      IF(VLK(JTH) .NE. KTH .OR. VTURN(JTH) .NE. 3)  GO TO 100
      IBNEW = IBHOLD(LKDEST(KTH,3))
      IF(IBNEW .EQ. 0)  GO TO 75
      CALL HOLDIN(JTH,IBNEW,IB,0)
      GO TO 100
   75 CALL UPDATV(JTH,0,0)
      N0 = 0
C**** WRITE(6,1) JTH,IB,IB,N0
    1 FORMAT('   MOVEFF',4I5)
C
C     ..... NEXT LOOK AT LEFT TURNS  WHICH HAVE BEGUN TO MOVE
C           OR ARE READY TO START.
C
  100 LPOS = 0
C
C     ..... FIRST SQUARE.
C
      IB = IBASE + LEFTSQ(1,IARM)
      IF(BLKARY(IB) .EQ. 0)  GO TO 110
      JTH = BLKARY(IB)
      IF(VLK(JTH) .NE. KTH)  GO TO 107
      IBNEW = IBHOLD(LKDEST(KTH,1))
      IF(IBNEW .EQ. 0)  GO TO 105
      CALL HOLDIN(JTH,IBNEW,IB,0)
      GO TO 130
  105 CALL UPDATV(JTH,0,0)
      N0 = 0
C**** WRITE(6,1) JTH,IB,IB,N0
      LPOS = 1
      GO TO 130
  107 LPOS = 1
C
C     ..... CHECK SECOND SQUARE.
C
  110 IB = IBASE + LEFTSQ(2,IARM)
      IF(BLKARY(IB) .EQ. 0)  GO TO 130
      JTH = BLKARY(IB)
      IF(VLK(JTH) .NE. KTH)  GO TO 125
      IF(LPOS .EQ. 1)  GO TO 105
      IF(VASP(JTH) .GT. 0)  GO TO 120
  115 IBNEW = IBASE + LEFTSQ(1,IARM)
      BLKARY(IB) = 0
      BLKARY(IBNEW) = JTH
      CALL UPDATV(JTH,1,0)
      N1 = 1
C**** WRITE(6,1) JTH,IB,IBNEW,N1
      LPOS = 1
      GO TO 130
  120 IBNEW = IBHOLD(LKDEST(KTH,1))
      IF(IBNEW .EQ. 0)  GO TO 115
      CALL HOLDIN(JTH,IBNEW,IB,1)
```

```
      GO TO 130
  125 LPOS = 1
C
C     ..... NOW CHECK TURNING POINT.
C
  130 IB = IBASE + LEFTSQ(3,IARM)
      IF(BLKARY(IB) .EQ. 0)  GO TO 150
      JTH = BLKARY(IB)
      IF(VLK(JTH) .NE. KTH .OR. VTURN(JTH) .NE. 1)  GO TO 150
      IF(LPOS .EQ. 1)  GO TO 145
      CALL GAPCHK(LKOPOS(KTH),KOUT,NTH,ITH,2-MOD(IARM,2))
      IF(KOUT .EQ. 1)  GO TO 145
C
C     ..... CHECK FIRST TWO SQUARES OF INTERSECTION FOR TRAFFIC
C           FROM OPPOSING LINK.
C
      IBCHK = IBASE + LEFTSQ(4,IARM)
      JTHCHK = BLKARY(IBCHK)
      IF(JTHCHK .EQ. 0)  GO TO 140
      IF(VLK(JTHCHK) .EQ. LKOPOS(KTH))  GO TO 145
  140 IBCHK = IBASE + LEFTSQ(5,IARM)
      JTHCHK = BLKARY(IBCHK)
      IF(JTHCHK .EQ. 0)  GO TO 142
      IF(VLK(JTHCHK) .EQ. LKOPOS(KTH))  GO TO 145
C
C     ..... OK TO MOVE VEHICLE INTO TURNING AREA.
C
  142 NUM = VASP(JTH) + 1
      IBNEW = IBASE + LEFTSQ(3-NUM,IARM)
      BLKARY(IB) = 0
      BLKARY(IBNEW) = JTH
C**** WRITE(6,1) JTH,IB,IBNEW,NUM
      CALL UPDATV(JTH,NUM,0)
      GO TO 150
  145 CALL UPDATV(JTH,0,0)
      NO = 0
C**** WRITE(6,1) JTH,IB,IB,NO
  150 CONTINUE
C
C     ..... LEFT IN MAIN FLOW TREATED WITH STRAIGHT TRAFFIC.
C
      DO 400 LANE=1,2
      JTHL = 0
      DO 400 POS=4,1,-1
      IB = IBASE + STRTSQ(POS,LANE,IARM)
      IF(BLKARY(IB) .EQ. 0)  GO TO 400
      JTH = BLKARY(IB)
      IF(VLK(JTH) .NE. KTH)  GO TO 390
      IF(JTHL)  380,210,250
C
C     ..... NO LEADER FOUND.
C
  210 IF(VTURN(JTH)-2) 211,215,400
C
C     ..... LEFT TURN VEHICLES.
C
  211 IF(POS .EQ. 3)  GO TO 212
```

```
      NUM = 1
      GO TO 220
  212 JTHL = JTH
      LPOS = POS
      GO TO 400
C
C     ..... STRAIGHT VEHICLES.
C
  215 NUM = VASP(JTH)
      IF(NUM .LT. VDSP(JTH) .AND. VSTATE(JTH) .NE. 1) NUM = NUM + 1
      IF(NUM+POS .GT. 4)  GO TO 225
  220 IBNEW = IBASE + STRTSO(POS+NUM,LANE,IARM)
      BLKARY(IB) = 0
      BLKARY(IBNEW) = JTH
      CALL UPDATV(JTH,NUM,0)
C**** WRITE(6,1) JTH,IB,IBNEW,NUM
      JTHL = JTH
      LPOS = POS
      GO TO 400
  225 IBNEW = IBHOLD(LKDEST(KTH,2))
      IF(IBNEW .NE. 0)  GO TO 230
      NUM = 4 - POS
      GO TO 220
  230 CALL HOLDIN(JTH,IBNEW,IB,4-POS)
      GO TO 400
C
C     ..... LEADER IS PRESENT.
C
  250 NGAP = LPOS - POS - 1
      IF(NGAP .EQ. 0)  GO TO 380
      IF(VTURN(JTH)-2)  211,260,400
  260 CALL ADVCHK(JTH,JTHL,NGAP,NUM)
      IF(NUM .EQ. 0)  GO TO 380
      GO TO 220
  380 JTHL = JTH
      LPOS = POS
      NO = 0
C**** WRITE(6,1) JTH,IB,IB,NO
      CALL UPDATV(JTH,0,0)
      GO TO 400
C
C     ..... LEADER IS PRESENT FROM ANOTHER LINK.
C
  390 LPOS = POS
      JTHL = -1
  400 CONTINUE
 1000 CONTINUE
      RETURN
      END
      SUBROUTINE MOVETT(NTH)
C
C     ..... THIS SUBROUTINE MOVES VEHICLES WHICH ARE IN A
C           TWO BY TWO INTERSECTION.  YOU ARE LOOKING AT
C           THE NORTH WEST SQUARE OF THE INTERSECTION AND
C           THE ARMS ARE LABELED AS SHOWN BELOW:
C
C
```

```
C                                          *    *
C                                          *  4 *
C                                          *    *
C                                    *  *  *    *  *  *
C
C                                       1          3
C
C                                    *  *  *    *  *  *
C                                          *    *
C                                          *  2 *
C                                          *    *
C
C
      INCLUDE DEFINS
C
C     ..... LOCAL ARRAYS
C
      DIMENSION IARM(4,4),IB13(4)
      DATA IB13/ 2,-1,1,-2 /
      DATA IARM/1,4,2,3,2,1,3,4,3,2,4,1,4,3,1,2 /
C
C     ..... SET UP A BASE FOR REFERENCING THE BLOCK ARRAY.
C
      IBASE = NOFSTB(NTH) - 1
C
C     ..... CONSIDER EACH SQUARE OF INTERSECTION
C
      DO 2000 ISQUAR=1,4
C
C     ..... FIND OUT IF THERE IS A VEHICLE IN THAT SQUARE, AND
C           IF NOT GO TO NEXT SQUARE.
      IB = IBASE + ISQUAR
      JTH = BLKARY(IB)
      IF (JTH .EQ. 0)  GO TO 2000
      IF(VMOVE(JTH) .EQ. 1)  GO TO 2000
C
C     ..... FIND OUT WHICH ARM VEHICLE IS FROM AND BRANCH.
C
      KTHOLD = VLK(JTH)
      I3ARM = IARM(ISQUAR,3)
      IF(NOINP(NTH,I3ARM) .EQ. KTHOLD) GO TO 1001
      I4ARM = IARM(ISQUAR,4)
      IF(NOINP(NTH,I4ARM) .EQ. KTHOLD) GO TO 1500
      I2ARM = IARM(ISQUAR,2)
      IF(NOINP(NTH,I2ARM) .EQ. KTHOLD) GO TO 1300
C
C     ..... IF THESE TESTS FAIL AN ERROR HAS OCCURRED
C
      CALL DUMP(JTH,'MOVETT','NOINP?',57)
C
C
C     ..... VEHICLE HAS COME FROM INPUT 3
C
 1001 IF(VTURN(JTH) .EQ. 1)  GO TO 1200
C
C     ..... INPUT 3 STRAIGHT TRAFFIC
      KTHNEW = LKDEST(KTHOLD,2)
```

```
C
C     ..... CHECK HOLDING SPACE
      IBNEW = LKFSTB(KTHNEW) + LKLNG(KTHNEW)
C
C     ..... IF HOLDING AREA NOT AVAILABLE, GO TO 1975.
      IF(BLKARY(IBNEW) .NE. 0)  GO TO 1975
C
C     ..... OTHERWISE MOVE TO HOLDING AREA.
      NOMOVE = 0
      GO TO 1960
C
C     ..... INPUT 3 LEFT TURN.  IF SQUARE 3 NON EMPTY, NO MOVE POSSIBLE.
 1200 IBCHK = IB13(ISQUAR) + IB
      IF(BLKARY(IBCHK) .NE. 0) GO TO 1975
C
C     ..... NEXT CHECK APPOSING TRAFFIC.  SKIP CHECK IF NO LINK.
      KTHOPO = LKOPOS(KTHOLD)
      IF(KTHOPO .EQ. 0)  GO TO 1250
      CALL GAPCHK(KTHOPO,KOUT,NTH,NOSIG(NTH),2-MOD(I3ARM,2))
C
C     ..... IF GAP NOT BIG ENOUGH, WAIT
      IF(KOUT .NE. 0)  GO TO 1975
C
C     ..... NEXT CHECK HOLDING AREA.
 1250 KTHNEW = LKDEST(KTHOLD,1)
      IBNEW = LKFSTB(KTHNEW) + LKLNG(KTHNEW)
C
C     ..... IF HOLDING AREA NOT AVAILABLE, WAIT
      IF(BLKARY(IBNEW) .NE. 0)  GO TO 1975
C
C     ..... OK TO TURN.  ADVANCE TO SQUARE 3.
      GO TO 1965
C
C
C     ..... VEHICLE HAS COME FROM INPUT 2 AND IS TURNING LEFT.
C
 1300 KTHNEW = LKDEST(KTHOLD,1)
C
C     ..... CHECK HOLDING AREA.
      IBNEW = LKFSTB(KTHNEW) + LKLNG(KTHNEW)
C
C     ..... IF HOLDING AREA BLOCKED, WAIT.
      IF(BLKARY(IBNEW) .NE. 0)  GO TO 1975
C
C     .....OTHERWISE MOVE TO HOLDING AREA.
      NOMOVE = 0
      GO TO 1960
C
C
C     ..... VEHICLE HAS COME FROM INPUT 4, BRANCH ON TURNING MOVEMENT.
 1500 K = VTURN(JTH)
      GO TO (1700,1600,1550), K
C
C     ..... INPUT 4 RIGHT TURN.  CHECK HOLDING AREA
 1550 KTHNEW = LKDEST(KTHOLD,3)
      IBNEW = LKFSTB(KTHNEW) + LKLNG(KTHNEW)
C
```

```
C      ..... IF AREA BLOCKED, WAIT.
       IF(BLKARY(IBNEW) .NE. 0)  GO TO 1975
C
C      ..... MOVE TO HOLDING AREA.
       NOMOVE = 0
       GO TO 1960
C
C      ..... INPUT 4 STRAIGHT MOVEMENT. CHECK SQUARE 3 AND SPEED.
 1600  IBCHK = IB13(ISQUAR) + IB
       IF(BLKARY(IBCHK) .NE. 0)  GO TO 1975
       IF(VASP(JTH) .LT. 1)  GO TO 1965
C
C      ..... NEXT CHECK HOLDING AREA.
       KTHNEW = LKDEST(KTHOLD,2)
       IBNEW = LKFSTR(KTHNEW) + LKLNG(KTHNEW)
C
C      ..... IF AREA BLOCKED, ADVANCE TO SQUARE 3
       IF(BLKARY(IBNEW) .NE. 0)  GO TO 1965
C
C      ..... MOVE VEHICLE THROUGH ONE BLOCK TO HOLDING AREA.
       NOMOVE = 1
       GO TO 1960
C
C      ..... INPUT 4 LEFT TURN.  CHECK SQUARE 3.
 1700  IBCHK = IB13(ISQUAR) + IB
       IF(BLKARY(IBCHK) .NE. 0)  GO TO 1975
       GO TO 1965
C
C      ..... OK TO MOVE VEHICLE TO HOLDING AREA OF DESTINATION.
C
 1960  CALL HOLDIN(JTH,IBNEW,IB,NOMOVE)
       GO TO 2000
C
C      ..... ADVANCE ONE SQUARE COUNTERCLOCKWISE.
C
 1965  IBNEW=IB+IB13(ISQUAR)
       BLKARY(IB)=0
       BLKARY(IBNEW)=JTH
       CALL UPDATV(JTH,1,0)
       N1 = 1
C**** WRITE(6,1) JTH,IB,IBNEW,N1
       GO TO 2000
C
C      ..... NO MOVEMENT WAS POSSIBLE.
C
 1975  CALL UPDATV(JTH,0,0)
       N0 = 0
C**** WRITE(6,1) JTH,IB,IB,N0
1      FORMAT(' MOVETT',4I6)
 2000  CONTINUE
       RETURN
       END
       FUNCTION RN1(ISEED)
C
C      ..... THIS FUNCTION GENERATES UNIFORMLY DISTRIBUTED
C            RANDOM NUMBERS BETWEEN 0 AND 1
C
```

```
      ISEED=ISEED*185333
      IF(ISEED .LT. 0) ISEED=ISEED+34359738367+1
      RN1=ISEED*2.0**(-35)
      RETURN
      END
      SUBROUTINE SETUP
C
C     ..... THIS SUBROUTINE DOES THE FINAL SETUP ON ALL DATA
C           FILES BEFORE THE SIMULATION BEGINS.
C
      INCLUDE DEFINS
C
C     ..... FOR EACH NODE
      DO 2000 NTH=1,NUMNOS
      IF(NOTYPE(NTH) .GT. 2) GO TO 1500
      IF(NOTYPE(NTH) .EQ. 2) GO TO 1300
C
C     ..... GENERATE NODES, SET TIME OF FIRST ARRIVAL.
      NOPARS(NTH,1) = 3600. / NOPARS(NTH,1)
      DO 1001 I=1,100
 1001 A = RN1(ISEED)
      NOSEED(NTH) = ISEED
      NOCLK(NTH) = TBAGEN(NTH)
C
C     ..... FIND LINK TO PUT VEHICLES ON.
      DO 1100 KTH=1,NUMLKS
      IF(NONXT(NTH,1) .EQ. LKID(KTH,2) .AND. NOIDNO(NTH) .EQ. LKID(KTH,1
     1))    GO TO 1110
 1100 CONTINUE
      CALL DUMP(KTH,'SETUP ','DO1100',1)
 1110 NOOUTP(NTH) = KTH
C
C     ..... IF GENERATE ONLY SKIP TO NEXT NODE.
C
      IF(NOTYPE(NTH) .EQ. 1) GO TO 2000
C
C     ..... FIND INDEX TO LINK POINTING AT THIS NODE.
C
 1300 CONTINUE
      DO 1350 KTH=1,NUMLKS
      IF(NONXT(NTH,1) .EQ. LKID(KTH,1) .AND. NOIDNO(NTH) .EQ. LKID(KTH,2))
     1))    GO TO 1360
 1350 CONTINUE
      CALL DUMP(KTH,'SETUP ','DO1350',1)
 1360 NOINP(NTH,1) = KTH
      GO TO 2000
C
C     ..... INTERNAL NODES
C
 1500 CONTINUE
      IDNTH = NOIDNO(NTH)
      ITH = NOSIG(NTH)
C
C     ..... TAKE EACH LINK AND SEE IF IT IS CONNECTED TO THIS
C           NODE.
C
      DO 1600 KTH=1,NUMLKS
```

```
C
C     ..... CHECK FOR INPUT TO THIS NODE.
      IF(LKID(KTH,2) .NE. IDNTH) GO TO 1600
C
C     ..... OK, NOW FIND OUT WHERE IT CAME FROM.
      DO 1520 I=1,4
      IF(LKID(KTH,1) .EQ. NONXT(NTH,I))  GO TO 1525
 1520 CONTINUE
      CALL DUMP(NTH,'SETUP ','D01520',1)
 1525 NOINP(NTH,I) = KTH
      IF(ITH .NE. 0)  SIGINP(ITH,I) = KTH
 1600 CONTINUE
 2000 CONTINUE
C
C     ..... FOR EACH LINK FIND THE DESTINATIONS.
      DO 3000 KTH=1,NUMLKS
      DO 2500 I=1,3
      IF(LKDEST(KTH,I) .EQ. 0)  GO TO 2500
      DO 2400 KTHCHK=1,NUMLKS
      IF(LKID(KTHCHK,2) .NE. LKDEST(KTH,I))  GO TO 2400
      IF(LKID(KTHCHK,1) .NE. LKID(KTH,2))  GO TO 2400
      LKDEST(KTH,I) = KTHCHK
      GO TO 2500
 2400 CONTINUE
 2500 CONTINUE
 3000 CONTINUE
      RETURN
      END
      SUBROUTINE SIGCHK
C
C     ..... THIS SUBROUTINE CHECKS THE SIGNAL SETTINGS TO
C           SEE IF A CHANGE IS NECESSARY.
C
      INCLUDE DEFINS
      DO 2000 ITH=1,NUMSIG
      DO 1999 I=1,2
C
C     ..... FIRST CHECK THE SIGNAL'S CLOCK.
C
      IF(SIGCLK(ITH,I) .GT. CLOCK)  GO TO 1999
C
C     ..... BRANCH BASED ON OLD STATE.
C
      IF(SIGSTA(ITH,I)-1)  1100,1200,1300
C
C     ..... OLD STATE WAS GREEN, CHANGE TO AMBER.
 1100 SIGSTA(ITH,I) = 1
      SIGCLK(ITH,I) = SIGCLK(ITH,I) + AMBERT
      GO TO 1999
C
C     ..... OLD STATE WAS AMBER, CHANGE TO RED.
 1200 SIGSTA(ITH,I) = 2
      SIGCLK(ITH,I) = SIGCLK(ITH,I) + SIGRED(ITH,I)
      GO TO 1999
C
C     ..... OLD STATE WAS RED, CHANGE TO GREEN.
 1300 SIGSTA(ITH,I) = 0
```

```
      SIGCLK(ITH,I) = SIGCLK(ITH,I) + SIGGRN(ITH,I)
C
C     ..... CHECK QUEUE LENGTHS ON INPUTS WHICH WERE RED.
C
      DO 1500 IP=1,2
      II = I + 2 * (IP - 1)
      KTH = SIGINP(ITH,II)
C
C     ..... CHECK FIRST LANE FOR QUEUE.
C
      IB1 = LKFSTB(KTH)
      IB2 = IB1 + LKLNG(KTH) -1
      LQ = 0
      DO 1420 IB=IB2,IB1,-1
      JTH = BLKARY(IB)
      IF(JTH .EQ. 0)  GO TO 1420
      IF(LQ .NE. 0)  GO TO 1410
      IF(VASP(JTH) .NE. 0)  GO TO 1420
 1410 LQ = LQ + 1
 1420 CONTINUE
      IF(LQ .GT. LKMAXQ(KTH,1))  LKMAXQ(KTH,1) = LQ
      IF(LKLANS(KTH) .EQ. 1)  GO TO 1500
C
C     ..... IF TWO LANES, CHECK THE SECOND LANE FOR QUEUE.
C
      IB1 = IB2 + 1
      IB2 = IB2 + LKLNG(KTH)
      LQ = 0
      DO 1470 IB=IB2,IB1,-1
      JTH = BLKARY(IB)
      IF(JTH .EQ. 0)  GO TO 1470
      IF(LQ .NE. 0)  GO TO 1460
      IF(VASP(JTH) .NE. 0)  GO TO 1470
 1460 LQ = LQ + 1
 1470 CONTINUE
      IF(LQ .GT. LKMAXQ(KTH,2))  LKMAXQ(KTH,2) = LQ
 1500 CONTINUE
 1999 CONTINUE
 2000 CONTINUE
C**** WRITE(6,1) CLOCK,((SIGSTA(I1,I2),I2=1,2),I1=1,NUMSIG)
    1 FORMAT(///,' CLOCK ',I9,'  SIGNALS:',20(1X,2I2))
      RETURN
      END
      FUNCTION SPDDIS(ISEED)
      DIMENSION ARY(4)
      DATA ARY / -1., 0.07, 0.97, 1.0 /
      X=RN1(ISEED)
      DO 100 I=2,4
      IF(ARY(I) .GE. X)  GO TO 200
  100 CONTINUE
  200 SPDDIS = I
      RETURN
      END
      SUBROUTINE STAT(K)
C
C     ..... THIS SUBROUTINE CLEARS THE STATISTICS AFTER WARM-UP
C           AND THEN PRINTS THEM AFTER THE FINISH.
```

```
C
      INCLUDE DEFINS
C
C     ..... IF FINISH, GO TO 1000
      IF(K .EQ. 2)  GO TO 1000
      DO 200 KTH = 1,NUMLKS
      DO 100 I=1,2
      LKMAXQ(KTH,I) = 0
      LKSTOP(KTH,I) = 0
      LKDEL(KTH,I) = 0
  100 LKVL(KTH,I) = 0
      DO 200 I=1,3
      LKDELD(KTH,I) = 0
  200 LKVD(KTH,I) = 0
      RETURN
C
C     ..... PRINT THE STATISTICS IN THIS SECTION.
C
 1000 TOTIM = FINISH - WARMUP
      WRITE(6,1)
    1 FORMAT(1H1,21X,'VEHICLE COUNT',15X,'VEHICLE VOLUME',21X,
     1  'VEHICLE DELAY',14X,'DELAY PER VEHICLE',/,' LINK   LANES   ',
     2  'LANE 1   LANE 2   TOTAL     LANE 1   LANE 2   TOTAL',
     3  12X,'LANE 1   LANE 2   TOTAL       LANE 1   LANE 2   TOTAL',/)
      DO 1100 KTH=1,NUMLKS
      LKVTOT = LKVL(KTH,1) + LKVL(KTH,2)
      LKVOL1 = LKVL(KTH,1) * 3600 / TOTIM
      LKVOL2 = LKVL(KTH,2) * 3600 / TOTIM
      LKVOLT = LKVOL1 + LKVOL2
      LKDELT = LKDEL(KTH,1) + LKDEL(KTH,2)
      DELVL1 = LKDEL(KTH,1) / FLOAT(LKVL(KTH,1))
      DELVL2 = LKDEL(KTH,2) / FLOAT(LKVL(KTH,2))
      DELVLT = LKDELT / FLOAT(LKVL(KTH,1) + LKVL(KTH,2))
 1100 WRITE(6,2) KTH,LKLANS(KTH),LKVL(KTH,1),LKVL(KTH,2),LKVTOT,
     1  LKVOL1,LKVOL2,LKVOLT,LKDEL(KTH,1),LKDEL(KTH,2),LKDELT,
     2  DELVL1,DELVL2,DELVLT
    2 FORMAT(I4,4X,I2,2(1X,3I9),9X,3I9,2X,3F9.2)
      WRITE(6,3)
    3 FORMAT(///,14X,'VEHICLE STOPS',11X,'MAX QUEUES',14X,'TURNS',
     1  19X,'TURN DELAY',14X,'TURN DELAY PER VEHICLE',/,
     2  ' LINK   LANE 1   LANE 2   TOTAL    LANE 1   LANE 2    LEFT',
     3  '  STRAIGHT RIGHT      LEFT    STRAIGHT  RIGHT       LEFT',
     4  '   STRAIGHT  RIGHT',/)
      DO 1200 KTH=1,NUMLKS
      LKSTOT = LKSTOP(KTH,1) + LKSTOP(KTH,2)
      DELVL1 = LKDELD(KTH,1) / LKVD(KTH,1)
      DELVL2 = LKDELD(KTH,2) / LKVD(KTH,2)
      DELVL3 = LKDELD(KTH,3) / LKVD(KTH,3)
 1200 WRITE(6,4) KTH,LKSTOP(KTH,1),LKSTOP(KTH,2),LKSTOT,LKMAXQ(KTH,1),
     1  LKMAXQ(KTH,2),(LKVD(KTH,I),I=1,3),(LKDELD(KTH,I),I=1,3),
     2  DELVL1,DELVL2,DELVL3
    4 FORMAT(I4,3(4X,I5),6X,I2,7X,I2,2X,3(3X,I5),2(4X,I6),3X,I6,
     1  2X,3(1X,F8.2))
    5 FORMAT(//,23H INTERSECTION ID NUMBER,I4,/,12H0INPUT LINKS,6X,
     1    14HVEHICLE VOLUME,10X,9HMAX QUEUE,11X,13HVEHICLE DELAY,12X,
     2    17HDELAY PER VEHICLE,/,14X,22HLANE 1  LANE 2   TOTAL,4X,
     3    14HLANE 1  LANE 2,2(4X,22HLANE 1  LANE 2   TOTAL),/)
```

```
    6 FORMAT(I7,4X,2I8,I9,2I8,3X,2I8,I9,2X,2F8.2,F9.2)
    7 FORMAT(30X,6H------,64X,6H------,/,21H  TOTAL INTERSECTION ,
     1   8HVOLUME = ,I7,23X,40HAVERAGE INTERSECTION DELAY PER VEHICLE =,
     2   F7.2)
    8 FORMAT(/////,32H AVERAGE LINK DELAY FOR SYSTEM = F7.2)
    9 FORMAT(29H1INTERSECTION OUTPUT SUMMARY.)
      WRITE(6,9)
      NTOT = 0
      NDTOT = 0
      DO 1275 NTH=1,NUMNOS
      IF(NOTYPE(NTH) .LT. 3)  GO TO 1275
      NINT = 0
      NDEL = 0
      WRITE(6,5) NOIDNO(NTH)
      ITH = NOSIG(NTH)
      DO 1250 II=1,4
      KTH = SIGINP(ITH,II)
      LKVTOT = LKVL(KTH,1) + LKVL(KTH,2)
      NINT = NINT + LKVTOT
      LKDELT = LKDEL(KTH,1) + LKDEL(KTH,2)
      NDEL = NDEL + LKDELT
      DELVL1 = LKDEL(KTH,1) / FLOAT(LKVL(KTH,1))
      DELVL2 = LKDEL(KTH,2) / FLOAT(LKVL(KTH,2))
      DELVLT = LKDELT / FLOAT(LKVTOT)
 1250 WRITE(6,6) KTH,LKVL(KTH,1),LKVL(KTH,2),LKVTOT,LKMAXQ(KTH,1),
     1           LKMAXQ(KTH,2),LKDEL(KTH,1),LKDEL(KTH,2),LKDELT,
     2           DELVL1,DELVL2,DELVLT
      NTOT = NTOT + NINT
      NDTOT = NDTOT + NDEL
      DEL = NDEL / FLOAT(NINT)
      WRITE(6,7) NINT,DEL
 1275 CONTINUE
      DEL = NDTOT / FLOAT(NTOT)
      WRITE(6,8) DEL
      RETURN
      END
      SUBROUTINE STATA(I)
C
C     ..... PERIOD BY PERIOD OUTPUT GENERATOR.  LKSOUT IS AN ARRAY
C           TELLING THE DESIRED OUTPUT LINKS.
C
      INCLUDE DEFINS
      DIMENSION LKSOUT(4),N1(4),AVG1(4),X1(4),L1(4),LL1(4),LS1(4),LR1(4)
     1   ,DEL1(4),NL1(4),NS1(4),NR1(4),DELY1(4)
      DATA LKSOUT / 1,5,0,0 /
      PARAMETER N=2
      GO TO (10,20),I
   10 CONTINUE
C**** WRITE(6,4)
      DO 15 II=1,N
      L1(II)=0
      DEL1(II)=0
      LL1(II)=0
      LS1(II)=0
   15 LR1(II)=0
      RETURN
C**** WRITE(6,1) CLOCK,NUMVEH
```

```
   20 CONTINUE
      DO 30 II=1,N
      MM = LKSOUT(II)
      L1P = LKVL(MM,1) + LKVL(MM,2)
      N1(II)=L1P - L1(II)
      L1(II)=L1P
      NL1(II)=LKVD(MM,1)-LL1(II)
      LL1(II)=LKVD(MM,1)
      NS1(II)=LKVD(MM,2)-LS1(II)
      LS1(II)=LKVD(MM,2)
      NR1(II)=LKVD(MM,3)-LR1(II)
      LR1(II)=LKVD(MM,3)
      IDELP = LKDEL(MM,1) + LKDEL(MM,2)
      DELY1(II)=IDELP-DEL1(II)
      DEL1(II)=IDELP
      AVG1(II)=DELY1(II)/N1(II)
      X1(II)=IDELP/FLOAT(L1P)
C**** WRITE(6,2) L1P,(LKVD(MM,J),J=1,3),
C****1  IDELP,X1(II),N1(II),NL1(II),NS1(II),NR1(II),AVG1(II)
   30 CONTINUE
      PUNCH 3,CLOCK,NUMVEH,((N1(II),AVG1(II),X1(II)),II=1,N)
      RETURN
    1 FORMAT(/,9H CLOCK = ,I7,13H    NUMVEH = ,I7)
    2 FORMAT(10X,I6,3I5,I8,F10.2,I10,3I5,F8.2)
    3 FORMAT(I7,I5,4(I5,2F6.1))
    4 FORMAT(1H1,12X,10HCUMULATIVE,18X,7HCURRENT,7X,11HLAST PERIOD,/,
     1  13X,25HVOL   LF   ST   RT  DELAY,7X,3HMOE,7X,
     2  25HVOL   LF   ST   RT  DEL/V)
      END
      FUNCTION TBAGEN(NTH)
C
C     ..... THIS FUNCTION GENERATES THE TIME BETWEEN ARRIVALS
C           AT THE NTH NODE.  AN EXPONENTIAL TIME BETWEEN
C           ARRIVALS IS ASSUMED.
C
      INCLUDE DEFINS
      TBAGEN = -ALOG(RN1(NOSEED(NTH))) * NOPARS(NTH,1)
      RETURN
      END
      SUBROUTINE TERMIN(JTH,IB)
C
C     ..... THIS SUBROUTINE TERMINATES VEHICLES THAT ARE
C           LEAVING THE NETWORK.
C
      INCLUDE DEFINS
C
C     ..... FIRST ZERO BLOCK ON LINK.
      BLKARY(IB) = 0
C
C     ..... THEN GET COUNT FOR VOLUME CHECK.
C
      LANE = VLAN(JTH)
      KTH = VLK(JTH)
      LKVL(KTH,LANE) = LKVL(KTH,LANE) + 1
C
C     ..... NEXT ZERO DESIRED SPEED AND LINK NUMBER.
C
```

```
      VDSP(JTH) = 0
      VLK(JTH) = 0
      VHOLD(JTH) = 0
      VLAN(JTH) = 0
      VTURN(JTH) = 0
C
C     ..... FINALLY DECREASE NUMBER OF VEHICLES IN THE NETWORK.
C
      NUMVEH = NUMVEH - 1
C**** WRITE(6,1) JTH,IB
1     FORMAT(' VEHICLE',I4,' TERMINATED FROM BLOCK ',I4)
      RETURN
      END
      SUBROUTINE UPDATV(JTH,NOBLKM,IDUM)
C
C     ..... THIS SUBROUTINE UPDATES INFORMATION ON ARRAY FOR
C           EACH VEHICLE.  CAN BE USED TO GATHER STATISTICS
C           IF DESIRED.
C
      INCLUDE DEFINS
      VMOVE(JTH)=1
C
C     ..... DID VEHICLE DECELERATE, STAY AT SAME SPEED OR
C           ACCELERATE THIS TIME PERIOD.
C
      IF(VASP(JTH)-NOBLKM)  10,20,30
C
C     ..... ACCELERATION SECTION.
C
   10 VSTATE(JTH)=1
      VASP(JTH)=NOBLKM
      RETURN
C
C     ..... SAME SPEED.
C
   20 VSTATE(JTH)=3
      IF(NOBLKM .EQ. 0 .AND. IDUM .NE. 1)  VSTATE(JTH)=0
      RETURN
C
C     ..... DECELERATION SECTION.
C
   30 VSTATE(JTH)=2
      VASP(JTH)=NOBLKM
      IF(NOBLKM .NE. 0 .OR. IDUM .EQ. 1)  RETURN
C
C     ..... VEHICLE HAS DECELERATED TO A STOP.
C
      VSTATE(JTH) = 0
      KTH = VLK(JTH)
      LANE = VLAN(JTH)
      LKSTOP(KTH,LANE) = LKSTOP(KTH,LANE) + 1
      RETURN
      END
      SUBROUTINE VCLEAR
C
C     ..... THIS SUBROUTINE SETS MOVE FLAG FOR ALL VEHICLES.
C
```

```
      INCLUDE DEFINS
      DO 1000 JTH=1,MAXVEH
 1000 VMOVE(JTH) = 0
      RETURN
      END
      SUBROUTINE VEGEN
C
C     ..... THIS SUBROUTINE CHECKS THE CLOCK AT EACH GENERATE
C           NODE TO SEE IF IT IS TIME TO GENERATE ANOTHER
C           VEHICLE.
C
      INCLUDE DEFINS
C**** WRITE(6,2)
    2 FORMAT(' VEGEN')
      DO 2000 NTH=1,NUMNOS
      IF(NOTYPE(NTH) .GT. 1)  GO TO 2000
C
C     ..... CHECK TO SEE IF READY FOR ANOTHER VEHICLE
C
 1000 IF (CLOCK .LT. NOCLK(NTH))  GO TO 2000
C
C     ..... FIND OUT IF HOLDING AREA IS AVAILABLE.
      IBNEW = IBHOLD(NOOUTP(NTH))
      IF(IBNEW .EQ. 0)  GO TO 2000
C
C     ..... HOLDING AREA READY, FIND AN AVAILABLE VEHICLE.
C
 1105 CONTINUE
      DO 1110 JTH=1,MAXVEH
      IF(VDSP(JTH) .EQ. 0)  GO TO 1121
 1110 CONTINUE
      CALL DUMP(0,'VEGEN ','MAXVEH',27)
C
C     ..... NOW FILL IN HIS PARAMETERS.
C
 1121 VDSP(JTH) = SPDDIS(ISEED)
      VASP(JTH) = VDSP(JTH)
      VSTATE(JTH) = 3
      NUMVEH = NUMVEH + 1
C
C     ..... PLACE VEHICLE IN HOLDING AREA
C
C**** WRITE(6,1) JTH,I
1     FORMAT('   NEW VEHICLE, NUMBER ',I4,'  VDSP = ',I2)
      CALL HOLDIN(JTH,IBNEW,0,0)
C
C     .....COMPUTE TIME OF NEXT ARRIVAL
C
      NOCLK(NTH) = NOCLK(NTH) + TBAGEN(NTH)
C
C     ..... READY FOR ANOTHER.
C
      GO TO 1000
 2000 CONTINUE
      RETURN
      END
```

APPENDIX D

FORTRAN LISTING OF ZONE MODEL

```
DEFINS PROCEDURE
C
C      ..... SETUP FOR NODES.
C
       PARAMETER MAXNO=35
       REAL NOPARS,NOCLK
       COMMON /NODES/ NOTDNO(MAXNO),NOTYPE(MAXNO),NONXT(MAXNO,4),
      1  NOSEED(MAXNO),NOPARS(MAXNO,2),NOOUTP(MAXNO),
      2  NOSIG(MAXNO),NOCLK(MAXNO),NUMNOS
C
C      ..... SETUP FOR SIGNALS.
C
       PARAMETER MAXSIG=20
       INTEGER AMBERT,SIGINP,SIGCYC,SIGSTA,SIGOFF,SIGGRN,SIGRED,SIGCLK
       COMMON /SIG/ SIGCYC(MAXSIG),SIGSTA(MAXSIG,2),SIGOFF(MAXSIG,2),
      1  SIGGRN(MAXSIG,2),SIGRED(MAXSIG,2),SIGCLK(MAXSIG,2),
      2  SIGINP(MAXSIG,4),NUMSIG,AMBERT
C
C      ..... SETUP FOR LINKS.
C
       PARAMETER MAXLKS=100
       REAL LKPROB
       COMMON /LINKS/ LKTD(MAXLKS,2),LKLNG(MAXLKS),LKLANS(MAXLKS),
      1  LKFSTB(MAXLKS),LKNOD(MAXLKS),LKNTYP(MAXLKS),LKOPOS(MAXLKS),
      2  LKARM(MAXLKS),LKLEFT(MAXLKS),LKPROB(MAXLKS,2),LKDEST(MAXLKS,3),
      3  NUMLKS
C
C      ..... SETUP FOR ZONES.
C
       PARAMETER MAXBLK=2000
       COMMON /BLOCKS/ BLKARY(MAXBLK),LENGTH,NBLKS
       INTEGER BLKARY
C
C      ..... GENERAL COMMON VARIABLES.
C
       INTEGER CLOCK,DELT,FINISH,CAP,WARMUP
       COMMON CLOCK,ISEED,LIMDCH,DELT,FINISH,CAP,WARMUP,NUMVEH,ISPEED
C
C      ..... STAT SETUP.
C
       COMMON /STAT/ LKDEL(MAXLKS,2),LKMAXQ(MAXLKS,2),LKVL(MAXLKS,2)
C
C      *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C      *                                                             *
C      *      DEFINITION OF VARIABLES USED IN THE PROGRAM.           *
C      *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C      *                                                             *
C      *                                                             *
C      *                                                             *
C      *      AMBERT - AMBER TIME IN SECONDS.
C      *      BLKARY(IB) - ARRAY OF BLOCKS.                          *
C      *      CAP - THE CAPACITY OF A ZONE.
C      *      CLOCK - MASTER SIMULATION CLOCK (IN SECONDS).          *
C      *      DELT - THE TIME STEP (IN SECONDS).
C      *      FINISH - THE CUTOFF TIME (IN SECONDS).
C      *      ITH - INDEX USED TO POINT TO SPECIFIC SIGNAL.          *
```

```
C     *     ISEED - SEED TO THE RANDOM NUMBER GENERATOR.
C     *     ISPEED - AVERAGE SPEED IN NETWORK (FEET PER SECOND).
C     *     KTH - INDEX USED TO POINT TO SPECIFIC LINK.                *
C     *     LENGTH - LENGTH OF A ZONE (IN FEET).
C     *     LKARM(KTH) - 1 IF MAJOR APPROACH, 2 IF MINOR.
C     *     LKDEL(KTH,1-2) - CUMULATIVE LINK DELAY BY LANES.
C     *     LKDEST(KTH,1-3) - LINK DESTINATIONS: LEFT,STR,RIGHT.       *
C     *     LKFSTB(KTH) - LINK'S FIRST BLOCK.                          *
C     *     LKID(KTH,1-2) - SYMBOLIC I.D. OF TAIL(1) AND HEAD(2).      *
C     *     LKLANS(KTH) - NUMBER OF LANES.                             *
C     *     LKLEFT(KTH) - FLAG INDICATING A VEHICLE DELAYED
C     *          MAKING A LEFT TURN.
C     *     LKLNG(KTH) - LINK LENGTH (IN BLOCKS).                      *
C     *     LKMAXQ(KTH,1-2) - MAXIMUM QUEUE LENGTH FOUND BY LANES.
C     *     LKNOD(KTH) - INDEX TO NODE AT HEAD OF LINK.                *
C     *     LKOPOS(KTH) - LINK APPOSING A LEFT TURN.                   *
C     *     LKPROB(KTH,1-2) - CUMULATIVE PROB OF LEFT & STR. MOVE.     *
C     *     MAXBLK - MAXIMUM POSSIBLE NUMBER OF BLOCKS.                *
C     *     MAXLKS - MAXIMUM POSSIBLE NUMBER OF LINKS.                 *
C     *     MAXNO  - MAXIMUM POSSIBLE NUMBER OF NODES.                 *
C     *     MAXSIG - MAXIMUM POSSIBLE NUMBER OF SIGNALS.               *
C     *     NBLKS - NUMBER OF BLOCKS IN MODEL.                         *
C     *     NOCLK(NTH) - TIME OF NEXT ARRIVAL AT NTH NODE.
C     *     NOIDNO(NTH) - SYMBOLIC NODE IDENTIFIER.                    *
C     *     NONXT(NTH,1-4) - SYMBOLIC IDENTIFIERS OF ADJ. NODES.       *
C     *     NOOUTP(NTH) - INDEX TO LINKS TAKING FROM THIS NODE         *
C     *     NOPARS(NTH,1-2) - PARAMETERS FOR GENERATE NODES.           *
C     *     NOSIG(NTH) - INDEX TO SIGNAL FOR NTH NODE.                 *
C     *     NOTYPE(NTH) - NODE TYPE:                                   *
C     *                          0 - BOTH GENERATE AND TERMINATE,      *
C     *                          1 - GENERATE ONLY,                    *
C     *                          2 - TERMINATE ONLY,                   *
C     *                          4 - TWO WAY STOP,                     *
C     *                          5 - FIXED TIME CONTROLLED.            *
C     *     NTH - INDEX USED TO POINT TO SPECIFIC NODE.                *
C     *     NUMLKS - NUMBER OF LINKS IN MODEL.                         *
C     *     NUMNOS - NUMBER OF NODES.                                  *
C     *     NUMSIG - NUMBER OF SIGNALS.                                *
C     *     NUMVEH - NUMBER OF VEHICLES IN NETWORK.                    *
C     *     SIGCLK(ITH,1-2) - SIGNAL CLOCK.
C     *     SIGCYC(ITH) - CYCLE LENGTH OF ITH SIGNAL.
C     *     SIGGRN(ITH,1-2) - GREEN TIME.
C     *     SIGINP(ITH,1-4) - LINKS POINTING AT SIGNAL.
C     *     SIGOFF(ITH,1-2) - OFFSET FOR SIGNAL.
C     *     SIGRED(ITH,1-2) - RED TIME.
C     *     SIGSTA(ITH,1-2) - STATE OF THE SIGNAL.
C     *     WARMUP - TIME SPENT BEFORE GATHERING STATISTICS.
C     *                                                                *
C     *                                                                *
C     *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C     *                                                                *
C     *                                                                *
C     *                                                                *
C     *                                                                *
C     *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C
C
```

```
C
 END
C
C      ..... THIS IS THE MAIN PROGRAM FOR THE ZONE MODEL.
C            THE CLOCK IS IN UNITS OF SECONDS.
C
C
      INCLUDE DEFINS,LIST
C
C      ..... INITIALIZATION
      NTH = 0
      NBLKS = 1
      ISPEED = 53
C
C      ..... RUN PARAMETERS
C
      READ(5,1) IRUN,INET,AMBERT,ISEED,DELT,FINISH,WARMUP
    1 FORMAT( )
      MODEL = 'ZONE'
      WRITE(22) MODEL,IRUN,INET,ISEED
      PUNCH 6,IRUN,INET,ISEED
    6 FORMAT('ZONE MODEL  IRUN = ',I4,'  INET = ',I4,'  ISEED = ',I12)
      LIMDCH = DELT / 2
      IF(LIMDCH .LE. 0)  LIMDCH = 1
      WARMUP = WARMUP * 60
      FINISH = FINISH * 60 + WARMUP
      CLOCK = -DELT
      LENGTH = ISPEED * DELT
      CAP = LENGTH / 20.0 + 0.5
      WRITE(6,3)
    3 FORMAT('1**************************   Z O N E   M O D E L',
     1  '   **************************',//)
      WRITE(6,2) IRUN,INET,ISEED,DELT,FINISH,WARMUP,ISPEED,LENGTH,CAP
    2 FORMAT(' RUN NUMBER ',I5,//,'     NETWORK ',I5,'      ISEED ',
     1  I9,//,'     DELT   ',I8,/,'     FINISH ',I8,/,'     WARMUP ',
     2  I8,//,'     ISPEED ',I3,'   ZONE LENGTH ',I4,'     ZONE CAPACITY',
     3  I3,///)
C      ..... CALL INPUT SUBROUTINES.
C
      CALL EXTERN(NTH)
      CALL INTERN(NTH)
      CALL INPLKS
C
C      ..... CALL SETUP TO FINISH DATA MANIPULATION.
C
      CALL SETUP
      CALL DUMP(0,'DATA  ','CHECK ',0)
C
C      ..... THE SIMULATION.
C
 1000 CLOCK = CLOCK + DELT
      IF(MOD(CLOCK,90).EQ.0)  CALL STATA(2)
      WRITE(22) CLOCK,NUMVEH
      IF(CLOCK .GE. WARMUP) GO TO 2000
      CALL SIGCHK
      CALL VEGEN
      CALL INTERL
```

```
      CALL INTRAL
      CALL HOLDOU
      GO TO 1000
C
C     ..... WARMUP OVER, CLEAR STATISTICS.
C
 2000 CALL STAT(1)
      CALL STATA(1)
      GO TO 2300
 2100 CLOCK = CLOCK + DELT
      IF(MOD(CLOCK,90).EQ.0)  CALL STATA(2)
      WRITE(22) CLOCK,NUMVEH
      IF(CLOCK .GE. FINISH) GO TO 3000
 2300 CALL SIGCHK
      CALL VEGEN
      CALL INTERL
      CALL INTRAL
      CALL HOLDOU
      GO TO 2100
C
C     ..... SIMULATION OVER, PRINT STATISTICS.
C
 3000 CALL STAT(2)
      ENDFILE 22
      STOP
      END
      SUBROUTINE DUMP(M,N1,N2,K)
C
C     ..... THIS SUBROUTINE PROVIDES A DUMP OF THE VARIABLES
C           IN CASE OF AN ERROR.  THE VALUE OF K DETERMINES
C           THE TYPE OF DUMP:
C                              0 - SET UP CHECK, CALL RETURN
C                                  AFTER PRINTOUT ON NODES, ETC.
C                              1 - ERROR ON INPUT,
C                            > 1 - ERROR WHILE RUNNING - LINE NUMBER.
C
      INCLUDE DEFINS
C
C     ..... GENERAL INFORMATION DUMP.
      WRITE(6,1) K
1     FORMAT(///' ERROR DUMP OUTPUT.  K =',I5)
      IF(K .GE. 2)  GO TO 100
      WRITE(6,2) M,N1,N2
2     FORMAT(' CODE ',I3,' MESSAGE - ',2A6)
      GO TO 1001
  100 WRITE(6,3) M,N1,N2
3     FORMAT('  LINK OR SIG ',I5,' NOW BEING PROCESSED.',/,' MESSAGE -'
     1  ,1X,2A6)
C
C     ..... NODE OUTPUT.
C
 1001 MAX = MAXNO
      WRITE(6,1010) MAX,NUMNOS
 1010 FORMAT(///,' NODE OUTPUT          MAX = ',I2,12X,
     1  'NUMBER = ',I2,//,' INDEX    ID     NEXT NODE ID',
     2  '     TYPE    GEOM    INDEX LINKS IN     LINK OUT   SIGNAL',
     3  '    FSTB   CLOCK  PARAMETERS',//)
```

```
      DO 101 I=1,NUMNOS
  101 WRITE(6,1011) I,NOIDNO(I),(NONXT(I,K),K=1,4),NOTYPE(I),
     1  NOOUTP(I),NOSIG(I),NOCLK(I),
     2  (NOPARS(I,K),K=1,2)
 1011 FORMAT(I4,6X,I2,3X,4I3,6X,I2,5X,2X,2X,16X,7X,I2,8X,I2,5X,
     1  3X,3X,F5.0,2X,F5.0,1X,F5.0)
C
C     ..... SIGNAL OUTPUT
C
      MAX = MAXSIG
      LAM = AMBERT
      WRITE(6,1020) MAX,NUMSIG,LAM
 1020 FORMAT(///,' SIGNAL OUTPUT      MAX = ',I2,'      NUMBER = ',
     1  I2,'      AMBER TIME = ',I2,//,23X,'* * * *   MAJOR',
     2  '   * * * *',22X,'* * * *   MINOR   * * * *',/,
     3  ' INDEX  CYCLE',2(6X,'OFFSET  GREEN   RED    STATE CLOCK  NXT ',
     4  ))
      DO 102 I=1,NUMSIG
  102 WRITE(6,1021) I,SIGCYC(I),(SIGOFF(I,K),SIGGRN(I,K),SIGRED(I,K),
     1  SIGSTA(I,K),SIGCLK(I,K),K=1,2)
 1021 FORMAT(I4,5X,I3,2(8X,I3,5X,I3,4X,I3,5X,I2,3X,I5,4X,2X))
C
C     ..... LINK OUTPUT.
C
      MAX = MAXLKS
      WRITE(6,1030) MAX,NUMLKS
 1030 FORMAT(///,' LINK OUTPUT      MAX = ',I3,'     NUMBER = ',I3,//,
     1  ' INDEX    ID NO.  LENGTH  LANES   NODE    LK OPOS    LK',
     2  ' DESTINATIONS    PROBABILITIES   FSTB  ARM  LEFT',//)
      DO 103 I=1,NUMLKS
  103 WRITE(6,1031) I,LKID(I,1),LKID(I,2),LKLNG(I),LKLANS(I),LKNOD(I),
     1  LKOPOS(I),(LKDEST(I,K),K=1,3),(LKPROB(I,K),K=1,2),LKFSTB(I),
     2  LKARM(I),LKLEFT(I)
 1031 FORMAT(I5,4X,2I3,I7,6X,I1,6X,I2,7X,I3,2I6,3X,I3,2X,
     1  2F7.3,5X,I4,3X,I1,5X,I1)
      IF(K .EQ. 0)  RETURN
      IF(K .EQ. 1) STOP
C
C     ..... BLOCK OUTPUT
C
      MAX = MAXBLK
      WRITE(6,1040) MAX,NBLKS
 1040 FORMAT('1BLOCK OUTPUT     MAX = ',I4,'     NUMBER = ',I4,//,
     1  ' FIRST * VALUE .....',/)
      I1 = -24
  104 I1 = I1 + 25
      I2 = I1 + 24
      WRITE(6,1041) I1,(BLKARY(II),II=I1,I2)
 1041 FORMAT(I5,'  *',25I4)
      IF(I2 .LT. NBLKS)  GO TO 104
      STOP
      END
      SUBROUTINE EXTERN(NTH)
C
C     ..... THIS SUBROUTINE READS THE DATA CARDS ON EXTERNAL
C           NODES ONLY.  CONTROL IS RETURNED TO THE CALLING
C           PROGRAM AFTER A BLANK CARD IS FOUND
```

```
C
      INCLUDE DEFINS
      INTEGER ALPHA
 1000 NTH=NTH+1
      READ(5,101) ALPHA,NOIDNO(NTH),NOTYPE(NTH),NONXT(NTH,1),
     1  NOPARS(NTH,1),NOPARS(NTH,2)
      IF(ALPHA .EQ. 1H )  GO TO 2000
      GO TO 1000
 2000 NTH=NTH-1
      RETURN
  101 FORMAT(A1,I3,I2,I3,2F5.0,I1)
      END
      SUBROUTINE HOLDOU
C
C     ..... THIS SUBROUTINE TAKES VEHICLES OUT OF A HOLDING
C           AREA AND MOVES THEM ONTO THE LINK.
C
      INCLUDE DEFINS
C
C**** WRITE(6,1)
    1 FORMAT(///,' HOLDOU SUBROUTINE')
      DO 100 KTH=1,NUMLKS
C
      IB = LKFSTB(KTH) + LKLNG(KTH) * LKLANS(KTH)
      IF(LKLANS(KTH) .EQ. 2)  GO TO 35
C
C     ..... SINGLE LANE STREET.
C
      IF(BLKARY(IB) .EQ. 0)  GO TO 100
      IBP = IB - 1
      IF(BLKARY(IB)+BLKARY(IBP)-CAP)  10,20,30
C
C     ..... NOT FULL.
   10 LKVL(KTH,1) = LKVL(KTH,1) + BLKARY(IB)
      BLKARY(IBP) = BLKARY(IBP) + BLKARY(IB)
      BLKARY(IB) = 0
C**** WRITE(6,2) KTH,IB,BLKARY(IB),IBP,BLKARY(IBP)
    2 FORMAT(' LINK ',I4,'  BLK ',I4,' CONT ',I4,'  BLK ',I4,
     1  ' CONT ',I4)
      GO TO 100
C
C     ..... FULL.
   20 LKVL(KTH,1) = LKVL(KTH,1) + BLKARY(IB)
      BLKARY(IBP) = CAP
      BLKARY(IB) = 0
C**** WRITE(6,2) KTH,IB,BLKARY(IB),IBP,BLKARY(IBP)
      GO TO 100
C
C     ..... OVERFLOW BACK INTO HOLDING AREA.
   30 LKVL(KTH,1) = LKVL(KTH,1) + CAP - BLKARY(IBP)
      BLKARY(IB) = BLKARY(IB) + BLKARY(IBP) - CAP
      BLKARY(IBP) = CAP
C**** WRITE(6,2) KTH,IB,BLKARY(IB),IBP,BLKARY(IBP)
      GO TO 100
C
C
C     ..... TWO LANE STREET, MUST CHECK BOTH AREAS.
```

```
C
   35 IBP = LKFSTB(KTH) + LKLNG(KTH) - 1
      IF(BLKARY(IB) .EQ. 0)  GO TO 65
      IF(BLKARY(IB)+BLKARY(IBP)-CAP)  40,50,60
C
C     ..... NOT FULL.
   40 LKVL(KTH,1) = LKVL(KTH,1) + BLKARY(IB)
      BLKARY(IBP) = BLKARY(IBP) + BLKARY(IB)
      BLKARY(IB) = 0
C**** WRITE(6,2) KTH,IB,BLKARY(IB),IBP,BLKARY(IBP)
      GO TO 65
C
C     ..... FULL.
   50 LKVL(KTH,1) = LKVL(KTH,1) + BLKARY(IB)
      BLKARY(IBP) = CAP
      BLKARY(IB) = 0
C**** WRITE(6,2) KTH,IB,BLKARY(IB),IBP,BLKARY(IBP)
      GO TO 65
C
C     ..... OVERFLOW BACK INTO HOLDING AREA.
   60 LKVL(KTH,1) = LKVL(KTH,1) + CAP - BLKARY(IBP)
      BLKARY(IB) = BLKARY(IB) + BLKARY(IBP) - CAP
      BLKARY(IBP) = CAP
C**** WRITE(6,2) KTH,IB,BLKARY(IB),IBP,BLKARY(IBP)
C
C     ..... SECOND HOLDING AREA OF TWO LANE STREET.
C
   65 IB = IB + 1
      IF(BLKARY(IB) .EQ. 0)  GO TO 100
      IBP = IB - 2
      IF(BLKARY(IB)+BLKARY(IBP)-CAP)  70,80,90
C
C     ..... NOT FULL.
   70 LKVL(KTH,2) = LKVL(KTH,2) + BLKARY(IB)
      BLKARY(IBP) = BLKARY(IBP) + BLKARY(IB)
      BLKARY(IB) = 0
C**** WRITE(6,2) KTH,IB,BLKARY(IB),IBP,BLKARY(IBP)
      GO TO 100
C
C     ..... FULL.
   80 LKVL(KTH,2) = LKVL(KTH,2) + BLKARY(IB)
      BLKARY(IBP) = CAP
      BLKARY(IB) = 0
C**** WRITE(6,2) KTH,IB,BLKARY(IB),IBP,BLKARY(IBP)
      GO TO 100
C
C     ..... OVERFLOW BACK INTO HOLDING AREA.
   90 LKVL(KTH,2) = LKVL(KTH,2) + CAP - BLKARY(IBP)
      BLKARY(IB) = BLKARY(IB) + BLKARY(IBP) - CAP
      BLKARY(IBP) = CAP
C**** WRITE(6,2) KTH,IB,BLKARY(IB),IBP,BLKARY(IBP)
  100 CONTINUE
      RETURN
      END
      FUNCTION IBHOLD(KTH)
C
C     ..... THIS FUNCTION RETURNS A 0 IF NO HOLDING
```

```
C           SPACE IS AVAILABLE.  OTHERWISE IT RETURNS THE
C           INDEX TO THE BLOCK ARRAY.
C
      INCLUDE DEFINS
C
      IBHOLD = 0
      IB = LKFSTB(KTH) + LKLNG(KTH) * LKLANS(KTH)
      IF(LKLANS(KTH) .EQ. 2)  GO TO 10
C
C     ..... SINGLE LANE STREET.
C
      IF(BLKARY(IB) .GE. CAP)  RETURN
    5 IBHOLD = IB
      RETURN
C
C     ..... TWO LANE STREET.
C
   10 IF(RN1(ISEED) .LT. 0.5)  GO TO 20
      IF(BLKARY(IB) .LT. CAP)  GO TO 5
      IB = IB + 1
      IF(BLKARY(IB) .LT. CAP)  GO TO 5
      RETURN
   20 IB = IB + 1
      IF(BLKARY(IB) .LT. CAP)  GO TO 5
      IB = IB - 1
      IF(BLKARY(IB) .LT. CAP)  GO TO 5
      RETURN
      END
      FUNCTION IGAP(KTH)
C
C     ..... THIS FUNCTION TESTS THE GAP THE VEHICLE IS OBSERVING.
C           A 0 MEANS GO AND A 1 MEANS STOP.
C
      INCLUDE DEFINS
      DIMENSION PROB(20)
      DATA PROB / -1., -1., .02, .10, .22, .36, .50, .61, .74,
     1    .81, .84, .90, .92, .95, .96, .97, .98, .99, .995, .9995  /
      L = LKLNG(KTH)
      IF(LKLANS(KTH) .EQ. 2)  GO TO 20
C
C     ..... CHECK SINGLE LANE.
C
      IF(LKLEFT(KTH) .EQ. 1)  GO TO 70
      IB2 = LKFSTB(KTH) - 1
    5 DO 10 I=1,L
      IB2 = IB2 + 1
      IF(BLKARY(IB2) .NE. 0)  GO TO 50
   10 CONTINUE
      GO TO 70
C
C     ..... CHECK TWO LANE STREETS.
C
   20 IB1 = LKFSTB(KTH) -1
      IB2 = IB1 + L
      IF(LKLEFT(KTH) .EQ. 1)  GO TO 5
      DO 30 I=1,L
      IB1 = IB1 + 1
```

```
      IB2 = IB2 + 1
      IF(BLKARY(IB1) .NE. 0 .OR. BLKARY(IB2) .NE. 0)  GO TO 50
   30 CONTINUE
      GO TO 70
   50 GAP = (I-1) * DELT
      IF(GAP .GE. 20.0)  GO TO   70
      IF(GAP .LE. 2)  GO TO   60
      INDEX = GAP  + 0.5
      IF(RN1(ISEED) .LE. PROB(INDEX))  GO TO   70
C
C     ..... REJECT.
   60 IGAP = 1
      RETURN
C
C     ..... ACCEPT.
   70 IGAP  = 0
      RETURN
      END
      SUBROUTINE INPLKS
C
C     ..... THIS SUBROUTINE READS CARDS FOR ALL NETWORK LINKS.
C           AFTER ALL DATA CARDS HAVE BEEN READ, THE ROUTINE
C           CALCULATES SOME CROSS REFERENCE INDICES.
C
      INCLUDE DEFINS
C
C     ..... THIS IS THE LOCAL DECLARATION.
C
      INTEGER ALPHA
      KTH=0
 1000 KTH=KTH+1
      READ(5,101) ALPHA,LKID(KTH,1),LKID(KTH,2),LKLNG(KTH),LKLANS(KTH),
     1  LKOPOS(KTH),LKPROB(KTH,1),LKPROB(KTH,2),(LKDEST(KTH,K),K=1,3)
C
C     ..... IF IT IS A BLANK CARD, GO TO 2000.
      IF(ALPHA .EQ. 1H )  GO TO 2000
C
C     ..... CALCULATE THE BLOCKS PER LANE AND THE NUMBER OF
C           BLOCKS USED THUS FAR.
      LKLNG(KTH)=LKLNG(KTH)/FLOAT(LENGTH) + 0.5
      IF(LKLNG(KTH) .LT. 1)  LKLNG(KTH) = 1
      LKFSTB(KTH)=NBLKS
      NBLKS=NBLKS+LKLNG(KTH)*LKLANS(KTH)+LKLANS(KTH)
C
C     ..... FIND NODE NUMBER OF THE HEAD NODE.
C
      DO 1010 NTH=1,NUMNOS
      IF(NODENO(NTH) .EQ. LKID(KTH,2))  GO TO 1015
 1010 CONTINUE
      CALL DUMP(KTH,'INPLKS','D01010',1)
 1015 LKNOD(KTH)=NTH
      DO 1016 I=1,4
      IF(LKID(KTH,1) .EQ. NONXT(NTH,I))  GO TO 1017
 1016 CONTINUE
      CALL DUMP(KTH,'INPLKS','D01016',1)
 1017 LKARM(KTH) = 2 - MOD(I,2)
C
```

```
C      ..... SET UP THE TURNING PROBABILITIES.
C
 1060 LKPROB(KTH,2)=LKPROB(KTH,1)+LKPROB(KTH,2)
      IF(LKPROB(KTH,2) .GE. 0.9985) LKPROB(KTH,2) = 1.0
      GO TO 1000
C
C      ..... FIND LINK OPPOSED.
C
 2000 NUMLKS=KTH-1
      IF(NUMLKS .GE. MAXLKS)  CALL DUMP(NUMLKS,'INPLKS',MAXLKS,1)
      IF(NBLKS .GT. MAXBLK)  CALL DUMP(NBLKS,'INPLKS','MAXBLK',1)
      DO 2500 KTH=1,NUMLKS
      IF(LKOPOS(KTH) .EQ. 0) GO TO 2500
      DO 2400 KTH2=1,NUMLKS
      IF(LKID(KTH2,2) .NE. LKID(KTH,2))  GO TO 2400
      IF(LKOPOS(KTH) .EQ. LKID(KTH2,1))  GO TO 2450
 2400 CONTINUE
      CALL DUMP(KTH,'INPLKS','LKOPOS',1)
 2450 LKOPOS(KTH)=KTH2
 2500 CONTINUE
      RETURN
  101 FORMAT(A1,2I3,I4,I1,I3,2F3.3,3I3)
      END
      SUBROUTINE INTERL
C
C      ..... THIS SUBROUTINE MOVES VEHICLES FROM ONE LINK TO
C            THE NEXT.
C
      INCLUDE DEFINS
C
C**** WRITE(6,5)
      DO 200 KTH=1,NUMLKS
C
C      ..... SET UP INITIAL VARIABLES.
C
      LANE = LKLANS(KTH)
      NTH = LKNOD(KTH)
      IB = LKFSTB(KTH)
      IF(LANE .NE. 1)  GO TO 6
C
C      ..... CHECK FOR CARS TO MOVE.
C
      IF(BLKARY(IB) .EQ. 0)  GO TO 200
      GO TO 7
    6 IB2 = IB + LKLNG(KTH)
      IF(BLKARY(IB) .EQ. 0 .AND. BLKARY(IB2) .EQ. 0)  GO TO 200
C
C      ..... IF NON TERMINATE NODE, GO TO 15
    7 IF(NOTYPE(NTH) .GE. 3)  GO TO 15
C
C      ..... TERMINATE SECTION.
   10 NUMVEH = NUMVEH - BLKARY(IB)
      BLKARY(IB) = 0
C**** WRITE(6,3) KTH,IB
      IF(LANE .EQ. 1)  GO TO 200
      NUMVEH = NUMVEH - BLKARY(IB2)
      BLKARY(IB2) = 0
```

```
C**** WRITE(6,3) KTH,IB2
      GO TO 200
C
C     ..... CHECK SIGNAL STATE FIRST.
C
   15 ITH = NOSIG(NTH)
      I = LKARM(KTH)
      IF(SIGSTA(ITH,I)-1)  50,20,25
C
C     ..... AMBER LIGHT WILL PERMIT ONE LEFT TURN IF WAITING.
C
   20 IF(LKLEFT(KTH) .NE. 0)  GO TO 30
C
C     ..... STOPPING VEHICLES.
   25 LKDEL(KTH,1) = LKDEL(KTH,1) + BLKARY(IB) * DELT
C**** WRITE(6,4) KTH,IB
C
C     ..... CHECK FOR ANOTHER LANE.
      IF(LANE .EQ. 1)  GO TO 200
      LKDEL(KTH,2) = LKDEL(KTH,2) + BLKARY(IB2) * DELT
C**** WRITE(6,4) KTH,IB2
      GO TO 200
C
C
   30 KTHCHK = LKDEST(KTH,1)
      IBP = IBHOLD(KTHCHK)
      IF(IBP .EQ. 0)  GO TO 25
      LKLEFT(KTH) = 0
      BLKARY(IBP) = BLKARY(IBP) + 1
      BLKARY(IB) = BLKARY(IB) - 1
      LKDEL(KTH,1) = LKDEL(KTH,1) + BLKARY(IB) * DELT
C**** WRITE(6,1) KTH,IB,IBP
C**** WRITE(6,4) KTH,IB
C
C     ..... CHECK FOR ANOTHER LANE.
C
      IF(LANE .EQ. 1)  GO TO 200
      LKDEL(KTH,2) = LKDEL(KTH,2) + BLKARY(IB2) * DELT
C**** WRITE(6,4) KTH,IB2
      GO TO 200
C
C     ..... GREEN LIGHT, BRANCH ON NUMBER OF LANES.
C
   50 NOMOVD = 0
      IF(LANE .EQ. 2)  GO TO 100
C
C     ..... CHECK THE TURN FLAG.
      IF(LKLEFT(KTH) .NE. 0)  GO TO 60
C
   51 IF(BLKARY(IB) .EQ. 0)  GO TO 200
      IF(NOMOVD .GE. LIMDCH)  GO TO 65
C
C     ..... PICK A DESTINATION FOR THE VEHICLE.
      X = RN1(ISEED)
      IF(X .LE. LKPROB(KTH,1))  GO TO 60
      IF(X .LE. LKPROB(KTH,2))  GO TO 55
C     ..... RIGHT.
```

```
      KTHCHK = LKDEST(KTH,3)
      GO TO 70
C     ..... STRAIGHT.
   55 KTHCHK = LKDEST(KTH,2)
      GO TO 70
C     ..... LEFT.
   60 KTHCHK = LKDEST(KTH,1)
      IBP = IBHOLD(KTHCHK)
      IF(IGAP(LKOPOS(KTH)) .NE. 1 .AND. IBP .NE. 0)  GO TO 75
C
C     ..... LEFT TURN HANGUP.
C
      LKLEFT(KTH) = 1
   65 LKDEL(KTH,1) = LKDEL(KTH,1) + BLKARY(IB) * DELT
C**** WRITE(6,2) KTH,IB,KTHCHK
      GO TO 200
C
C     ..... CHECK HOLDING AREA.
C
   70 IBP = IBHOLD(KTHCHK)
      IF(IBP .EQ. 0)  GO TO 65
C
C     ..... MOVE VEHICLE INTO HOLDING AREA.
C
   75 LKLEFT(KTH) = 0
      BLKARY(IB) = BLKARY(IB) - 1
      BLKARY(IBP) = BLKARY(IBP) + 1
      NOMOVD = NOMOVD + 1
C**** WRITE(6,1) KTH,IB,KTHCHK
      GO TO 51
C
C
C     ..... TWO LANE STREET, TAKE LANES ONE AT A TIME.
C
  100 XTEST = LKPROB(KTH,1)*2
      IF(LKLEFT(KTH) .NE. 0)  GO TO 105
C
  101 IF(BLKARY(IB) .EQ. 0)  GO TO 150
      IF(NOMOVD .GE. LIMDCH)  GO TO 110
C
C     ..... PICK DESTINATION.
C
      IF(RN1(ISEED) .GT. XTEST)  GO TO 120
C
C     ..... LEFT.
  105 KTHCHK = LKDEST(KTH,1)
      IBP = IBHOLD(KTHCHK)
      IF(IGAP(LKOPOS(KTH)) .NE. 1 .AND. IBP .NE. 0)  GO TO 125
C
C     ..... MARK DELAY AND GO TO NEXT LANE.
C
      LKLEFT(KTH) = 1
  110 LKDEL(KTH,1) = LKDEL(KTH,1) + BLKARY(IB) * DELT
C**** WRITE(6,2) KTH,IB,KTHCHK
      GO TO 150
C
C     ..... STRAIGHT.
```

```
  120 KTHCHK = LKDEST(KTH,2)
      IBP = IBHOLD(KTHCHK)
      IF(IBP .EQ. 0)  GO TO 110
C
C     ..... MOVE INTO HOLDING AREA.
C
  125 LKLEFT(KTH) = 0
      BLKARY(IBP) = BLKARY(IBP) + 1
      BLKARY(IB) = BLKARY(IB) -1
      NOMOVD = NOMOVD + 1
C**** WRITE(6,1) KTH,IB,KTHCHK
      GO TO 101
C
C     ..... NOW CHECK THE SECOND LANE.
C
  150 XTEST = (1.0 - LKPROB(KTH,2)) * 2
      NOMOVD = 0
  151 IF(BLKARY(IB2) .EQ. 0)  GO TO 200
      IF(NOMOVD .GE. LIMDCH)  GO TO 170
C
C     ..... PICK A DESTINATION.
C
      IF(RN1(ISEED) .LT. XTEST)  GO TO 155
C
C     ..... STRAIGHT.
      KTHCHK = LKDEST(KTH,2)
      GO TO 160
C
C     ..... RIGHT.
  155 KTHCHK = LKDEST(KTH,3)
C
  160 IBP = IBHOLD(KTHCHK)
      IF(IBP .EQ. 0)  GO TO 170
      BLKARY(IBP) = BLKARY(IBP) + 1
      BLKARY(IB2) = BLKARY(IB2) - 1
      NOMOVD = NOMOVD + 1
C**** WRITE(6,1) KTH,IB2,KTHCHK
      GO TO 151
  170 LKDEL(KTH,2) = LKDEL(KTH,2) + BLKARY(IB2) * DELT
C**** WRITE(6,2) KTH,IB2,KTHCHK
  200 CONTINUE
    1 FORMAT(' LINK ',I4,'  MOVE VEH FROM BLK ',I4,' TO LK ,,I4)
    2 FORMAT(' LINK ',I4,' VEH IN ',I4,' BLOCKED FOR LINK ',I4)
    3 FORMAT(' LINK ',I4,' TERMINATE VEHICLES IN BLK ',I4)
    4 FORMAT(' LINK ',I4,' VEH IN BLK ',I4,' STOPPED FOR LIGHT.')
    5 FORMAT(//,' INTERI SUBROUTINE.')
      RETURN
      END
      SUBROUTINE INTERN(NTH)
C
C     ..... THIS SUBROUTINE READS DATA CARDS ON INTERIOR
C           NODES ONLY.  CONTROL IS RETURNED TO THE CALLING
C           PROGRAM AFTER A BLANK CARD IS FOUND.
C
      INTEGER ALPHA,TEMP
      INCLUDE DEFINS
      ITH = 0
```

```
 1000 NTH = NTH + 1
      READ(5,101) ALPHA,NOIDNO(NTH),NOTYPE(NTH),(NONXT(NTH,I),I=1,4)
      IF(ALPHA .EQ. 1H )   GO TO 3000
C
C     ..... IF NONSIGNALIZED RETURN TO READING NEXT NODE.
C
      IF(NOTYPE(NTH) .LE. 4)  GO TO 1000
C
C     ..... SIGNALIZED INTERSECTIONS REQUIRE ADDITIONAL CARD.
C
      ITH = ITH + 1
      NOSIG(NTH) = ITH
      READ(5,102) ALPHA,KDUM,SIGCYC(ITH),SIGOFF(ITH,1),SIGGRN(ITH,1),
     1  SIGOFF(ITH,2),SIGGRN(ITH,2)
C
C     ..... CALCULATE STATE CHANGE TIMES AND PUT ON CHAIN.
C
      DO 2100 I=1,2
      SIGRED(ITH,I) = SIGCYC(ITH) - SIGGRN(ITH,I) - AMBERT
      TEMP = SIGOFF(ITH,I)  + SIGGRN(ITH,I)
      IF(TEMP .GT. SIGCYC(ITH))  GO TO 2010
      TEMP = TEMP + AMBERT
      IF(TEMP .GT. SIGCYC(ITH))  GO TO 2005
      SIGSTA(ITH,I) = 2
      SIGCLK(ITH,I) = SIGOFF(ITH,I)
      GO TO 2100
 2005 SIGSTA(ITH,I) = 1
      SIGCLK(ITH,I) = TEMP - SIGCYC(ITH)
      GO TO 2100
 2010 SIGSTA(ITH,I) = 0
      SIGCLK(ITH,I) = TEMP - SIGCYC(ITH)
      GO TO 2100
 2100 CONTINUE
      GO TO 1000
 3000 NUMNOS = NTH - 1
      IF(NUMNOS .GE. MAXNO)  CALL DUMP(NUMNOS,'INTERN','MAXNO ',1)
      NUMSIG = ITH
      IF(NUMSIG .GE. MAXSIG)  CALL DUMP(NUMSIG,'INTERN','MAXSIG',1)
      RETURN
C
C     ..... FORMAT STATEMENTS
C
  101 FORMAT(A1,I3,I2,4I3,I1)
  102 FORMAT(A1,I3,5I3)
      END
      SUBROUTINE INTRAL
C
C     ..... THIS SUBROUTINE MOVES VEHICLES ON THE LINKS FROM
C           ONE ZONE TO THE NEXT.
C
      INCLUDE DEFINS
C
C**** WRITE(6,1)
    1 FORMAT(//,' INTRAL SUBROUTINE')
    2 FORMAT(' LINK ',I4,' LANE ',I4,' IBFST ',I4,' IBLST ',I4)
    3 FORMAT('   BLOCK ',I4,' CONT ',I4,' BLOCK ',I4,' CONT ',I4)
      DO 100 KTH=1,NUMLKS
```

```
 1000 NTH = NTH + 1
      READ(5,101) ALPHA,NOIDNO(NTH),NOTYPE(NTH),(NONXT(NTH,I),I=1,4)
      IF(ALPHA .EQ. 1H )   GO TO 3000
C
C     ..... IF NONSIGNALIZED RETURN TO READING NEXT NODE.
C
      IF(NOTYPE(NTH) .LE. 4)  GO TO 1000
C
C     ..... SIGNALIZED INTERSECTIONS REQUIRE ADDITIONAL CARD.
C
      ITH = ITH + 1
      NOSIG(NTH) = ITH
      READ(5,102) ALPHA,KDUM,SIGCYC(ITH),SIGOFF(ITH,1),SIGGRN(ITH,1),
     1  SIGOFF(ITH,2),SIGGRN(ITH,2)
C
C     ..... CALCULATE STATE CHANGE TIMES AND PUT ON CHAIN.
C
      DO 2100 I=1,2
      SIGRED(ITH,I) = SIGCYC(ITH) - SIGGRN(ITH,I) - AMBERT
      TEMP = SIGOFF(ITH,I)  + SIGGRN(ITH,I)
      IF(TEMP .GT. SIGCYC(ITH))  GO TO 2010
      TEMP = TEMP + AMBERT
      IF(TEMP .GT. SIGCYC(ITH))  GO TO 2005
      SIGSTA(ITH,I) = 2
      SIGCLK(ITH,I) = SIGOFF(ITH,I)
      GO TO 2100
 2005 SIGSTA(ITH,I) = 1
      SIGCLK(ITH,I) = TEMP - SIGCYC(ITH)
      GO TO 2100
 2010 SIGSTA(ITH,I) = 0
      SIGCLK(ITH,I) = TEMP - SIGCYC(ITH)
      GO TO 2100
 2100 CONTINUE
      GO TO 1000
 3000 NUMNOS = NTH - 1
      IF(NUMNOS .GE. MAXNO)  CALL DUMP(NUMNOS,'INTERN','MAXNO ',1)
      NUMSIG = ITH
      IF(NUMSIG .GE. MAXSIG)  CALL DUMP(NUMSIG,'INTERN','MAXSIG',1)
      RETURN
C
C     ..... FORMAT STATEMENTS
C
  101 FORMAT(A1,I3,I2,4I3,I1)
  102 FORMAT(A1,I3,5I3)
      END
      SUBROUTINE INTRAL
C
C     ..... THIS SUBROUTINE MOVES VEHICLES ON THE LINKS FROM
C           ONE ZONE TO THE NEXT.
C
      INCLUDE DEFINS
C
C**** WRITE(6,1)
    1 FORMAT(//,' INTRAL SUBROUTINE')
    2 FORMAT(' LINK ',I4,' LANE ',I4,' IBFST ',I4,' IBLST ',I4)
    3 FORMAT('   BLOCK ',I4,' CONT ',I4,' BLOCK ',I4,' CONT ',I4)
      DO 100 KTH=1,NUMLKS
```

```
C
C     ..... LINK SETUP.
C
      LANE = 0
      IF(LKLNG(KTH) .EQ. 1)  GO TO 100
      IBFST = LKFSTR(KTH) + 1
    5 IBLST = IBFST + LKLNG(KTH) - 2
      LANE = LANE + 1
C**** WRITE(6,2) KTH,LANE,IBFST,IBLST
C
C     ..... CHECK EACH ZONE STARTING WITH THE SECOND.
C
      DO 90 IB=IBFST,IBLST
      IBP = IB - 1
C
C     ..... CHECK CAPICITY.
C
      IF(BLKARY(IB) .EQ. 0)  GO TO 90
      IF(BLKARY(IB) + BLKARY(IBP) - CAP)  10,20,30
C
C     ..... NOT FULL.
   10 BLKARY(IBP) = BLKARY(IBP) + BLKARY(IB)
      BLKARY(IB) = 0
C**** WRITE(6,3) IB,BLKARY(IB),IBP,BLKARY(IBP)
      GO TO 90
C
C     ..... FULL.
   20 BLKARY(IBP) = CAP
      BLKARY(IB) = 0
C**** WRITE(6,3) IB,BLKARY(IB),IBP,BLKARY(IBP)
      GO TO 90
C
C     ..... OVERFLOW - DELAY.
   30 BLKARY(IB) = BLKARY(IB) + BLKARY(IBP) - CAP
      BLKARY(IBP) = CAP
      LKDEL(KTH,LANE) = LKDEL(KTH,LANE) + BLKARY(IB) * DELT
C**** WRITE(6,3) IB,BLKARY(IB),IBP,BLKARY(IBP)
C
   90 CONTINUE
C
C     ..... IS THIS THE LAST LANE.
C
      IF(LANE .EQ. LKLANS(KTH))  GO TO 100
C
      IBFST = IBLST + 2
      GO TO 5
C
  100 CONTINUE
      RETURN
      END
      FUNCTION RN1(ISEED)
C
C     ..... THIS FUNCTION GENERATES UNIFORMLY DISTRIBUTED
C           RANDOM NUMBERS BETWEEN 0 AND 1
C
      ISEED=ISEED*185333
      IF(ISEED .LT. 0) ISEED=ISEED+34359738367+1
```

```
      RN1=ISEED*2.0**(-35)
      RETURN
      END
      SUBROUTINE SETUP
C
C     ..... THIS SUBROUTINE DOES THE FINAL SETUP ON ALL DATA
C           FILES BEFORE THE SIMULATION BEGINS.
C
      INCLUDE DEFINS
C
C     ..... FOR EACH NODE
      DO 2000 NTH=1,NUMNOS
      IF(NOTYPE(NTH) .GT. 2) GO TO 1500
      IF(NOTYPE(NTH) .EQ. 2) GO TO 2000
C
C     ..... GENERATE NODES, SET TIME OF FIRST ARRIVAL.
      NOPARS(NTH,1) = 3600. / NOPARS(NTH,1)
      DO 1001 I=1,100
 1001 A = RN1(ISEED)
      NOSEED(NTH) = ISEED
      NOCLK(NTH) = TBAGEN(NTH)
C
C     ..... FIND LINK TO PUT VEHICLES ON.
      DO 1100 KTH=1,NUMLKS
      IF(NONXT(NTH,1) .EQ. LKID(KTH,2) .AND. NOIDNO(NTH) .EQ. LKID(KTH,1
     1))    GO TO 1110
 1100 CONTINUE
      CALL DUMP(NTH,'SETUP ',' 1100C',1)
 1110 NOOUTP(NTH) = KTH
      GO TO 2000
C
C     ..... INTERNAL NODES
C
 1500 CONTINUE
      ITH = NOSIG(NTH)
      IF(ITH .EQ. 0)  GO TO 2000
      IDNTH = NOIDNO(NTH)
C
C     ..... TAKE EACH LINK AND SEE IF IT IS CONNECTED TO THIS
C           NODE.
C
      DO 1600 KTH=1,NUMLKS
C
C     ..... CHECK FOR INPUT TO THIS NODE.
      IF(LKID(KTH,2) .NE. IDNTH) GO TO 1600
C
C     ..... OK, NOW FIND OUT WHERE IT CAME FROM.
      DO 1520 I=1,4
      IF(LKID(KTH,1) .EQ. NONXT(NTH,I))  GO TO 1525
 1520 CONTINUE
      CALL DUMP(NTH,'SETUP ',' 1520C',1)
 1525 SIGINP(ITH,I) = KTH
 1600 CONTINUE
 2000 CONTINUE
C
C     ..... FOR EACH LINK FIND THE DESTINATIONS.
      DO 3000 KTH=1,NUMLKS
```

```
      DO 2500 I=1,3
      IF(LKDEST(KTH,I) .EQ. 0)  GO TO 2500
      DO 2400 KTHCHK=1,NUMLKS
      IF(LKID(KTHCHK,2) .NE. LKDEST(KTH,I))  GO TO 2400
      IF(LKID(KTHCHK,1) .NE. LKID(KTH,2))  GO TO 2400
      LKDEST(KTH,I) = KTHCHK
      GO TO 2500
 2400 CONTINUE
 2500 CONTINUE
 3000 CONTINUE
      RETURN
      END
      SUBROUTINE SIGCHK
C
C     ..... THIS SUBROUTINE CHECKS THE SIGNAL SETTINGS TO
C           SEE IF A CHANGE IS NECESSARY.
C
      INCLUDE DEFINS
      DO 2000 ITH=1,NUMSIG
      DO 1999 I=1,2
C
C     ..... FIRST CHECK THE SIGNAL'S CLOCK.
C
      IF(SIGCLK(ITH,I) .GT. CLOCK)  GO TO 1999
C
C     ..... BRANCH BASED ON OLD STATE.
C
      IF(SIGSTA(ITH,I)-1)  1100,1200,1300
C
C     ..... OLD STATE WAS GREEN, CHANGE TO AMBER.
 1100 SIGSTA(ITH,I) = 1
      SIGCLK(ITH,I) = SIGCLK(ITH,I) + AMBERT
      GO TO 1999
C
C     ..... OLD STATE WAS AMBER, CHANGE TO RED.
 1200 SIGSTA(ITH,I) = 2
      SIGCLK(ITH,I) = SIGCLK(ITH,I) + SIGRED(ITH,I)
      GO TO 1999
C
C     ..... OLD STATE WAS RED, CHANGE TO GREEN.
 1300 SIGSTA(ITH,I) = 0
      SIGCLK(ITH,I) = SIGCLK(ITH,I) + SIGGRN(ITH,I)
C
C     ..... CHECK QUEUE LENGHT FOR THE APPROACHES THAT HAVE CHANGED.
C
      DO 1500 IP=1,2
      II = I + 2 * (IP - 1)
      KTH = SIGINP(ITH,II)
C
C     ..... SET UP AND CHECK FIRST LANE.
C
      IB1 = LKFSTB(KTH)
      IB2 = IB1 + LKLNG(KTH) -1
      LQ = 0
      DO 1400 IB=IB1,IB2
      IF(BLKARY(IB) .LT. CAP)  GO TO 1410
      LQ = LQ + CAP
```

```
 1400 CONTINUE
      IB = IB2 + 1
      IF(LKLNG(KTH) .EQ. 2)  IB = IB + LKLNG(KTH)
 1410 IF(BLKARY(IB) .LT. CAP / 2.0)  GO TO 1420
      LQ = LQ + BLKARY(IB)
 1420 IF(LQ .GT. LKMAXQ(KTH,1))  LKMAXQ(KTH,1) = LQ
      IF(LKLANS(KTH) .EQ. 1)  GO TO 1500
C
C     ..... IF TWO LANES, CHECK SECOND LANE ALSO.
C
      IB1 = IB2 + 1
      IB2 = IB2 + LKLNG(KTH)
      LQ = 0
      DO 1450 IB=IB1,IB2
      IF(BLKARY(IB) .LT. CAP)  GO TO 1460
      LQ = LQ + CAP
 1450 CONTINUE
      IB = IB2 + 2
 1460 IF(BLKARY(IB) .LT. CAP / 2.0)  GO TO 1470
      LQ = LQ + BLKARY(IB)
 1470 IF(LQ .GT. LKMAXQ(KTH,2))  LKMAXQ(KTH,2) = LQ
 1500 CONTINUE
 1999 CONTINUE
 2000 CONTINUE
C**** WRITE(6,1) CLOCK,((SIGSTA(I1,I2),I2=1,2),I1=1,NUMSIG)
    1 FORMAT(///,' CLOCK ',I9,'  SIGNALS:',20(1X,2I2))
      RETURN
      END
      SUBROUTINE STAT(K)
C
C     ..... THIS SUBROUTINE CLEARS THE STATISTICS AFTER WARMUP
C           AND THEN PRINTS THEM AFTER THE FINISH.
C
      INCLUDE DEFINS
C
C     ..... IF FINISH, GO TO 1000
      IF(K .EQ. 2)  GO TO 1000
      DO 100 KTH = 1,NUMLKS
      DO 100 I=1,2
      LKDEL(KTH,I) = 0
      LKMAXQ(KTH,I) = 0
  100 LKVL(KTH,I) = 0
      RETURN
C
C     ..... PRINT THE STATISTICS IN THIS SECTION.
C
 1000 TOTIM = FINISH - WARMUP
      WRITE(6,1)
    1 FORMAT(1H1,21X,'VEHICLE COUNT',15X,'VEHICLE VOLUME',7X,'MAX QS',
     1  9X,'VEHICLE DELAY',14X,'DELAY PER VEHICLE',/,' LINK   LANES   ',
     2  'LANE 1   LANE 2   TOTAL      LANE 1   LANE 2   TOTAL     1    ',
     3  '2   LANE 1   LANE 2   TOTAL       LANE 1   LANE 2   TOTAL',/)
      DO 1100 KTH=1,NUMLKS
      LKVTOT = LKVL(KTH,1) + LKVL(KTH,2)
      LKVOL1 = LKVL(KTH,1) * 3600 / TOTIM
      LKVOL2 = LKVL(KTH,2) * 3600 / TOTIM
      LKVOLT = LKVOL1 + LKVOL2
```

```
      LKDELT = LKDEL(KTH,1) + LKDEL(KTH,2)
      DELVL1 = LKDEL(KTH,1) / FLOAT(LKVL(KTH,1))
      DELVL2 = LKDEL(KTH,2) / FLOAT(LKVL(KTH,2))
      DELVLT = LKDELT / FLOAT(LKVL(KTH,1) + LKVL(KTH,2))
 1100 WRITE(6,2) KTH,LKLANS(KTH),LKVL(KTH,1),LKVL(KTH,2),LKVTOT,
     1  LKVOL1,LKVOL2,LKVOLT,(LKMAXQ(KTH,I),I=1,2),LKDEL(KTH,1),
     2  LKDEL(KTH,2),LKDELT,DELVL1,DELVL2,DELVLT
    2 FORMAT(I4,4X,I2,2(1X,3I9),3X,I3,2X,I3,I8,2I9,2X,3F9.2)
    5 FORMAT(//,23H INTERSECTION ID NUMBER,I4,/,12H0INPUT LINKS,6X,
     1   14HVEHICLE VOLUME,10X,9HMAX QUEUE,11X,13HVEHICLE DELAY,12X,
     2   17HDELAY PER VEHICLE,/,14X,22HLANE 1  LANE 2   TOTAL,4X,
     3   14HLANE 1  LANE 2,2(4X,22HLANE 1  LANE 2   TOTAL),/)
    6 FORMAT(I7,4X,2I8,I9,2I8,3X,2I8,I9,2X,2F8.2,F9.2)
    7 FORMAT(30X,6H------,64X,6H------,/,21H  TOTAL INTERSECTION ,
     1  8HVOLUME = ,I7,23X,40HAVERAGE INTERSECTION DELAY PER VEHICLE =,
     2  F7.2)
    8 FORMAT(/////,32H AVERAGE LINK DELAY FOR SYSTEM = F7.2)
    9 FORMAT(29H1INTERSECTION OUTPUT SUMMARY.)
      WRITE(6,9)
      NTOT = 0
      NDTOT = 0
      DO 1275 NTH=1,NUMNOS
      IF(NOTYPE(NTH) .LT. 3)  GO TO 1275
      NINT = 0
      NDEL = 0
      WRITE(6,5) NOIDNO(NTH)
      ITH = NOSIG(NTH)
      DO 1250 II=1,4
      KTH = SIGINP(ITH,II)
      LKVTOT = LKVL(KTH,1) + LKVL(KTH,2)
      NINT = NINT + LKVTOT
      LKDELT = LKDEL(KTH,1) + LKDEL(KTH,2)
      NDEL = NDEL + LKDELT
      DELVL1 = LKDEL(KTH,1) / FLOAT(LKVL(KTH,1))
      DELVL2 = LKDEL(KTH,2) / FLOAT(LKVL(KTH,2))
      DELVLT = LKDELT / FLOAT(LKVTOT)
 1250 WRITE(6,6) KTH,LKVL(KTH,1),LKVL(KTH,2),LKVTOT,LKMAXQ(KTH,1),
     1            LKMAXQ(KTH,2),LKDEL(KTH,1),LKDEL(KTH,2),LKDELT,
     2            DELVL1,DELVL2,DELVLT
      NTOT = NTOT + NINT
      NDTOT = NDTOT + NDEL
      DEL = NDEL / FLOAT(NINT)
      WRITE(6,7) NINT,DEL
 1275 CONTINUE
      DEL = NDTOT / FLOAT(NTOT)
      WRITE(6,8) DEL
      RETURN
      END
      SUBROUTINE STATA(I)
C
C     ..... PERIOD BY PERIOD OUTPUT GENERATOR.  LKSOUT IS AN ARRAY
C           TELLING THE DESIRED OUTPUT LINKS.
C
      INCLUDE DEFINS
      DIMENSION LKSOUT(4),N1(4),AVG1(4),X1(4),L1(4),LL1(4),LS1(4),LR1(4)
     1   ,DEL1(4),NL1(4),NS1(4),NR1(4),DELY1(4)
      DATA LKSOUT / 1,3,0,0 /
```

```
      PARAMETER N=2
      GO TO (10,20),I
   10 CONTINUE
C**** WRITE(6,4)
      DO 15 II=1,N
      NL1(II)=0
      NS1(II)=0
      NR1(II)=0
      L1(II)=0
      DEL1(II)=0
      LL1(II)=0
      LS1(II)=0
   15 LR1(II)=0
      RETURN
C**** WRITE(6,1) CLOCK,NUMVEH
   20 CONTINUE
      DO 30 II=1,N
      MM = LKSOUT(II)
      L1P = LKVL(MM,1) + LKVL(MM,2)
      N1(II)=L1P - L1(II)
      L1(II)=L1P
      IDELP = LKDEL(MM,1) + LKDEL(MM,2)
      DELY1(II)=IDELP-DEL1(II)
      DEL1(II)=IDELP
      AVG1(II)=DELY1(II)/N1(II)
      X1(II)=IDELP/FLOAT(L1P)
C**** WRITE(6,2) L1P,
C****1  IDELP,X1(II),N1(II),NL1(II),NS1(II),NR1(II),AVG1(II)
   30 CONTINUE
      PUNCH 3,CLOCK,NUMVEH,((N1(II),AVG1(II),X1(II)),II=1,N)
      RETURN
    1 FORMAT(/,9H CLOCK = ,I7,13H    NUMVEH = ,I7)
    2 FORMAT(10X,I6,3I5,I8,F10.2,I10,3I5,F8.2)
    3 FORMAT(I7,I5,4(I5,2F6.1))
    4 FORMAT(1H1,12X,10HCUMULATIVE,18X,7HCURRENT,7X,11HLAST PERIOD,/,
     1  13X,25HVOL   LF   ST   RT  DELAY,7X,3HMOE,7X,
     2  25HVOL   LF   ST   RT  DEL/V)
      END
      FUNCTION TBAGEN(NTH)
C
C     ..... THIS FUNCTION GENERATES THE TIME BETWEEN ARRIVALS
C           AT THE NTH NODE.  AN EXPONENTIAL TIME BETWEEN
C           ARRIVALS IS ASSUMED.
C
      INCLUDE DEFINS
      TBAGEN = -ALOG(RN1(NOSEED(NTH))) * NOPARS(NTH,1)
      RETURN
      END
      SUBROUTINE VEGEN
C
C     ..... THIS SUBROUTINE CHECKS THE CLOCK AT EACH GENERATE
C           NODE TO SEE IF IT IS TIME TO GENERATE ANOTHER
C           VEHICLE.
C
      INCLUDE DEFINS
C**** WRITE(6,1)
    1 FORMAT(//,' VEGEN SUBROUTINE.')
```

```
      DO 2000 NTH=1,NUMNOS
      IF(NOTYPE(NTH) .GT. 1)  GO TO 2000
C
C     ..... CHECK TO SEE IF READY FOR ANOTHER VEHICLE
C
 1000 IF (CLOCK .LT. NOCLK(NTH))  GO TO 2000
C
C     ..... FIND OUT IF HOLDING AREA IS AVAILABLE.
C
      KTH = NOOUTP(NTH)
      IB = IBHOLD(KTH)
      IF(IB .EQ. 0)  GO TO 2000
C
C     ..... MOVE A VEHICLE INTO THE AREA.
C
 1900 BLKARY(IB) = BLKARY(IB) + 1
      NUMVEH = NUMVEH + 1
C
C     .....COMPUTE TIME OF NEXT ARRIVAL
      NOCLK(NTH) = NOCLK(NTH) + TBAGEN(NTH)
C**** WRITE(6,2) NTH,KTH,IB,NUMVEH
    2 FORMAT(' NTH ',I4,' KTH ',I4,' IB ',I4,' NUMVEH ',I5)
C
C     ..... READY FOR ANOTHER.
      GO TO 1000
 2000 CONTINUE
      RETURN
      END
```

APPENDIX E

FORTRAN LISTING OF NEXT MODEL

```
DEFINS PROCEDURE
C
C      ..... SETUP FOR NODES.
C
       PARAMETER MAXNO=35
       REAL NOPARS,NOCLK
       INTEGER FSTSG
       COMMON /NODES/ NOTONO(MAXNO),NOTYPE(MAXNO),NONXT(MAXNO,4),
      1  NOSEED(MAXNO),NOGNXT(MAXNO),NOPARS(MAXNO,2),NOOUTP(MAXNO),
      2  NOGEOM(MAXNO),FSTSG(MAXNO),NOSIG(MAXNO),NOCLK(MAXNO),NUMNOS
C
C      ..... SETUP FOR SIGNALS.
C
       PARAMETER MAXSIG=20
       INTEGER SIGCYC,SIGGRN,SIGRED,SIGOFF,SIGNXT,SIGSTA,SIGCLK,SIGINP
      1    ,AMBERT
       COMMON /SIG/ SIGCYC(MAXSIG),SIGSTA(MAXSIG,2),SIGOFF(MAXSIG,2),
      1  SIGGRN(MAXSIG,2),SIGRED(MAXSIG,2),SIGCLK(MAXSIG,2),
      2  SIGNXT(MAXSIG,2),SIGINP(MAXSIG,4),NUMSIG,AMBERT
C
C      ..... SETUP FOR LINKS.
C
       PARAMETER MAXLKS=100
       REAL LKPROB
       COMMON /LINKS/ LKTO(MAXLKS,2),LKLNG(MAXLKS),LKLANS(MAXLKS),
      1  LKNOD(MAXLKS),LKOPOS(MAXLKS),LKARM(MAXLKS),LKCAP(MAXLKS),
      2  LKCONT(MAXLKS),LKPROB(MAXLKS,2),LKDEST(MAXLKS,3),NUMLKS,
      3  LKDEL(MAXLKS,2),LKDELD(MAXLKS,3),LKQ(MAXLKS,2),LKQCLK(MAXLKS,2),
      4  LKSTOP(MAXLKS,2),LKVL(MAXLKS,2),LKVD(MAXLKS,3),LKMAXQ(MAXLKS,2),
      5  LKLEFT(MAXLKS),LKARVT(MAXLKS,2),LKDTS(MAXLKS,2)
C
C      ..... SETUP FOR VEHICLES.
C
       PARAMETER MAXVEH=2000
       INTEGER VLKTIM,VLK,VLAN,VTURN,VDSP,VTIME,VSTATE,VEHNXT
       COMMON /VEH/ VDSP(MAXVEH),VLK(MAXVEH),VLAN(MAXVEH),VTURN(MAXVEH),
      1  VTIME(MAXVEH),VSTATE(MAXVEH),VEHNXT(MAXVEH),VLKTIM(MAXVEH),
      2  NUMVEH
C
C      ..... SETUP FOR INTERSECTIONS.
C
       PARAMETER MAXBLK=200
       INTEGER BLKARY
       COMMON /INT/ BLKARY(MAXBLK),NBLKS
C
C      ..... SETUP FOR CHAINS.
C
       INTEGER SIGTIM,SIGIND,VEGTIM,VEGIND,VEGFLG
       COMMON /CHAIN/ VEGTIM,VEGIND,VEGFLG,MOVFST,MOVTIM,SIGTIM,SIGIND
C
C      ..... GENERAL COMMON VARIABLES.
C
       INTEGER CLOCK,FINISH,WARMUP
       COMMON CLOCK,ISEED,FINISH,WARMUP
C
C      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
C     *                                                               *
C     *     DEFINITION OF VARIABLES USED IN THE PROGRAM.              *
C     *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C     *                                                               *
C     *                                                               *
C     *                                                               *
C     *     AMBERT - THE LENGTH OF AMBER TIME.
C     *     BLKARY(IB) - GENERAL ARRAY OF INTERSECTION BLOCKS.
C     *     CLOCK - MASTER SIMULATION CLOCK.                          *
C     *     FINISH - INITIALLY THE AMOUNT OF TIME TO GATHER STATISTICS.
C     *           THEN CHANGED TO SIMULATION FINISH TIME.
C     *     FSTSQ(NTH) - INDEX TO FIRST SQUARE OF INTERSECTION.
C     *     ISEED - THE RANDOM NUMBER SEED.
C     *     ITH - INDEX USED TO POINT TO SPECIFIC SIGNAL.             *
C     *     KTH - INDEX USED TO POINT TO SPECIFIC LINK.               *
C     *     LKARM(KTH) - APPROACH ARM (1-4) OF LINK TO INTERSECTION.
C     *     LKCAP(KTH) - CAPACITY OF THE KTH LINK IN VEHICLES.
C     *     LKCONT(KTH) - CONTENTS OF THE KTH LINK.
C     *     LKDEL(KTH,1-2) - CUMULATIVE DELAY BY LANES.
C     *     LKDELD(KTH,1-3) - CUMULATIVE DELAY BY DESTINATIONS.
C     *     LKDEST(KTH,1-3) - LINK DESTINATIONS: LEFT,STR,RIGHT.      *
C     *     LKID(KTH,1-2) - SYMBOLIC I.D. OF TAIL(1) AND HEAD(2).     *
C     *     LKLANS(KTH) - NUMBER OF LANES.                            *
C     *     LKLEFT(KTH) - FLAG INDICATING AN EXISTING LEFT TURN DELAY.
C     *     LKLNG(KTH) - LINK LENGTH (IN FEET ).                      *
C     *     LKMAXQ(KTH,1-2) - CURRENT MAX QUEUE BY LANES.
C     *     LKNOD(KTH) - INDEX TO NODE AT HEAD OF LINK.               *
C     *     LKOPOS(KTH) - LINK APPOSING A LEFT TURN.                  *
C     *     LKPROB(KTH,1-2) - CUMULATIVE PROB OF LEFT & STR. MOVE.    *
C     *     LKQ(KTH,1-2) - CURRENT QUEUE LENGTH BY LANES.
C     *     LKQCLK(KTH,1-2) - TIME OF LAST DEPARTURE.
C     *     LKSTOP(KTH,1-2) - CUMULATIVE NUMBER OF STOPS.
C     *     LKVD(KTH,1-3) - LINK VEHICLE COUNT BY DESTINATION.
C     *     LKVL(KTH1-2) - LINK VEHICLE COUNT BY LANES.
C     *     MAXBLK - MAXIMUM NUMBER OF INTERSECTION BLOCKS.
C     *     MAXLKS - MAXIMUM POSSIBLE NUMBER OF LINKS.                *
C     *     MAXNO  - MAXIMUM POSSIBLE NUMBER OF NODES.                *
C     *     MAXSIG - MAXIMUM POSSIBLE NUMBER OF SIGNALS.              *
C     *     MAXVEH - MAXIMUM POSSIBLE NUMBER OF VEHICLES.             *
C     *     MOVFST - FIRST VEHICLE ON THE MOVE CHAIN.
C     *     MOVTIM - EVENT TIME FOR NEXT VEHICLE MOVEMENT.
C     *     NBLKS - NUMBER OF INTERSECTION BLOCKS IN NETWORK.
C     *     NOCLK(NTH) - ARRIVAL TIME OF NEXT VEHICLE AT NODE.
C     *     NOGEOM(NTH) - NODE GEOMETRY: 1 - 2X2, 3 - 4X4.
C     *     NOGNXT(NTH) - NEXT NODE ON THE GENERATE CHAIN.
C     *     NOIDNO(NTH) - SYMBOLIC NODE IDENTIFIER.                   *
C     *     NONXT(NTH,1-4) - SYMBOLIC IDENTIFIERS OF ADJ. NODES.      *
C     *     NOOUTP(NTH) - INDEX TO LINKS TAKING FROM THIS NODE        *
C     *     NOPARS(NTH,1-2) - PARAMETERS FOR GENERATE NODES.          *
C     *     NOSEED(NTH) - RANDOM NUMBER SEED FOR GENERATION.
C     *     NOSIG(NTH) - INDEX TO SIGNAL FOR NTH NODE.                *
C     *     NOTYPE(NTH) - NODE TYPE:                                  *
C     *                         0 - BOTH GENERATE AND TERMINATE,      *
C     *                         1 - GENERATE ONLY,                    *
C     *                         2 - TERMINATE ONLY,                   *
C     *                         5 - FIXED TIME CONTROLLED.            *
C     *     NTH - INDEX USED TO POINT TO SPECIFIC NODE.               *
```

```
C      *      NUMLKS -.NUMBER OF LINKS IN MODEL.                          *
C      *      NUMNOS - NUMBER OF NODES.                                   *
C      *      NUMSIG - NUMBER OF SIGNALS.                                 *
C      *      NUMVEH - NUMBER OF VEHICLES IN NETWORK.                     *
C      *      SIGCLK(ITH,1-2) - TIME OF NEXT CHANGE OF STATE.
C      *      SIGCYC(ITH) - CYCLE LENGTH OF ITH SIGNAL.
C      *      SIGGRN(ITH,1-2) - GREEN TIME FOR ITH SIGNAL.
C      *      SIGIND - POINTER TO NEXT SIGNAL TO CHANGE.
C      *      SIGINP(ITH,1-4) - LINKS POINTING AT ITH SIGNAL.
C      *      SIGNXT(ITH,1-2) - NEXT SIGNAL ON THE EVENTS CHAIN,
C      *                NEGATIVE INDICATES MINOR DIRECTION.
C      *      SIGOFF(ITH,1-2) - SIGNAL OFFSET TIME.
C      *      SIGRED(ITH,1-2) - RED TIME FOR ITH SIGNAL.
C      *      SIGSTA(ITH,1-2) - STATE OF ITH SIGNAL:
C      *.               0 - GREEN,
C      *                1 - AMBER,
C      *                2 - RED.
C      *      SIGTIM - TIME OF NEXT SIGNAL CHANGE.
C      *      VDSP(JTH) - VEHICLE'S DESIRED SPEED.                        *
C      *      VEGFLG - FLAG INDICATING A DELAYED ENTRY TO PROCESS.
C      *      VEGIND - INDEX TO FIRST NODE ON THE GENERATE CHAIN.
C      *      VEGTIM - EVENT TIME OF THE NEXT UNBLOCKED ARRIVAL.
C      *      VEHNXT(JTH) - NEXT VEHICLE ON MOVEMENT CHAIN.
C      *      VLAN(JTH) - LANE DESIRED.                                   *
C      *      VLK(JTH) - INDEX OF CURRENT OR MOST RECENT LINK.            *
C      *      VLKTIM(JTH) - CLOCK TIME VEHICLE ENTERED LINK.
C      *      VSTATE(JTH) - THE STATE OF THE JTH VEHICLE:
C      *                0 - ON LINK ARRIVING AT INTERSECTION,
C      *                1 - IN QUEUE WAITING (BUT NOT AT HEAD)
C      *                2 - AT HEAD OF QUEUE WAITING FOR SIGNAL,
C      *                3 - AT HEAD OF QUEUE IN BLOCKED STATE,
C      *                4 - IN S1,
C      *                5 - IN S2,
C      *                6 - IN S3.
C      *      VTIME(JTH) - NEXT EVENT TIME FOR JTH VEHICLE.
C      *      VTURN(JTH) - TURN INDICATOR:                                *
C      *                          1 - LEFT,                               *
C      *                          2 - STRAIGHT,                           *
C      *                          3 - RIGHT.                              *
C      *      WARMUP - WARM UP TIME TO GAIN STATISTICAL EQUIL.
C      *                                                                  *
C      *                                                                  *
C      *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C      *                                                                  *
C      *                                                                  *
C      *                                                                  *
C      *                                                                  *
C      *                                                                  *
C      *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C
C
C
 END
C
C      ..... THIS IS THE MAIN PROGRAM FOR THE NEXT EVENT MODEL.
C            ONE CLOCK UNIT EQUALS ONE TENTH OF A SECOND.
C
C
```

```
      INCLUDE DEFINS,LIST
C
C     ..... INITIALIZATION
      NTH = 0
      MOVTIM = 2**30
      NBLKS = 0
C
C     ..... RUN PARAMETERS
C
      READ(5,1) IRUN,INET,AMBERT,ISEED,FINISH,WARMUP
    1 FORMAT( )
      MODEL = 'NEXT'
      WRITE(21) MODEL,IRUN,INET,ISEED
      WRITE(22) MODEL,IRUN,INET,ISEED
      PUNCH 6,IRUN,INET,ISEED
    6 FORMAT('NEXT MODEL  IRUN = ',I4,'  INET = ',I4,'  ISEED = ',I12)
      CLOCK = -10
      AMBERT = AMBERT * 10
      WARMUP = WARMUP * 600
      FINISH = FINISH * 600 + WARMUP
      WRITE(6,2)
    2 FORMAT('1**********************   N E X T   M O D E L',
     1  '   **********************',///)
      WRITE(6,3) IRUN,INET,ISEED,FINISH,WARMUP
    3 FORMAT(' RUN NUMBER ',I5,//,'    NETWORK ',I5,'    ISEED ',
     1  I9,//,'    FINISH ',I8,'  WARMUP ',I8)
C
C     ..... CALL INPUT SUBROUTINES.
C
      CALL EXTERN(NTH)
      CALL INTERN(NTH)
      CALL INPLKS
C
C     ..... CALL SETUP TO FINISH DATA MANIPULATION.
C
      CALL SETUP
      CALL DUMP(0,'DATA C','HECK  ',0)
C
C     ..... THE SIMULATION.
C
 1000 ICLOCK = MIN(VEGTIM,SIGTIM,MOVTIM)
      IF(ICLOCK .LE. CLOCK)  CALL DUMP(ICLOCK,'ICLK =','CLK   ',CLOCK)
      CLOCK = ICLOCK
      IF(MOD(CLOCK,900) .EQ. 0)  CALL STATA(2)
      WRITE(22) CLOCK,NUMVEH
C**** IT1 = ITIME(IT2,IT3)
C**** XTIME = IT1 * .0002
C**** WRITE(6,4) CLOCK,NUMVEH,XTIME
    4 FORMAT(/,' CLOCK =',I6,' NUMVEH =',I4,' CPU TIME REMAINING ='F5.2)
      IF(CLOCK .GE. WARMUP)  GO TO 2000
      IF(CLOCK .LT. SIGTIM)  GO TO 1100
      CALL SIGCHK
 1100 IF(CLOCK .LT. VEGTIM)  GO TO 1200
      CALL VEGEN
 1200 IF(CLOCK .LT. MOVTIM) GO TO 1300
      CALL MOVVEH
      IF(VEGFLG .EQ. 1)  CALL VEGEN
```

```
 1300 GO TO 1000
C
C      ..... WARMUP OVER, CLEAR STATISTICS.
C
 2000 CALL STAT(1)
      CALL STATA(1)
      GO TO 2200
 2100 ICLOCK = MIN(VEGTIM,SIGTIM,MOVTIM)
      IF(ICLOCK .LE. CLOCK)  CALL DUMP(ICLOCK,'ICLK =','CLK    ',CLOCK)
      CLOCK = ICLOCK
      IF(MOD(CLOCK,900) .EQ. 0)  CALL STATA(2)
      WRITE(22) CLOCK,NUMVEH
C**** IT1 = ITIME(IT2,IT3)
C**** XTIME = IT1 * .0002
C**** WRITE(6,4) CLOCK,NUMVEH,XTIME
      IF(CLOCK .GT. FINTSH) GO TO 3000
 2200 CONTINUE
      IF(CLOCK .LT. SIGTIM)  GO TO 2300
      CALL SIGCHK
 2300 IF(CLOCK .LT. VEGTIM)  GO TO 2400
      CALL VEGEN
 2400 IF(CLOCK .LT. MOVTIM) GO TO 2100
      CALL MOVVEH
      IF(VEGFLG .EQ. 1)  CALL VEGEN
 2500 GO TO 2100
C
C      ..... SIMULATION OVER, PRINT STATISTICS.
C
 3000 CALL STAT(2)
      CALL CHAIN
      ENDFILE 21
      ENDFILE 22
      STOP
      END
      SUBROUTINE CHAIN
C
C      ..... THIS SUBROUTINE PRINTS THE VEHICLE CHAIN IN THE
C            PROPER SEQUENCE.
C
      INCLUDE DEFINS
      JTH = MOVFST
      WRITE(6,1) NUMVEH
   10 IF(JTH .EQ. 0)  GO TO 20
      WRITE(6,2) JTH,VEHNXT(JTH),VTIME(JTH),VDSP(JTH),VSTATF(JTH),
     1  VLK(JTH),VLAN(JTH),VTURN(JTH),VLKTIM(JTH)
      JTH = VEHNXT(JTH)
      GO TO 10
   20 WRITE(6,3) (LKCONT(KTH),KTH=1,NUMLKS)
      RETURN
    1 FORMAT('  VEHICLE CHAIN - NUMVEH',I5,/,'     JTH    NXT    TIME',
     1  '    DSP     STATF  LINK   LANE   TURN LKTIME')
    2 FORMAT(9(1X,I6))
    3 FORMAT('   LINK CONTENTS:',20I4)
      END
      SUBROUTINE DUMP(M,N1,N2,K)
C
C      ..... THIS SUBROUTINE PROVIDES A DUMP OF THE VARIABLES
```

```
C            IN CASE OF AN ERROR.  THE VALUE OF K DETERMINES
C            THE TYPE OF DUMP:
C                              0 - SET UP CHECK, CALL RETURN
C                                  AFTER PRINTOUT ON NODES, ETC.
C                              1 - ERROR ON INPUT,
C                            > 1 - ERROR WHILE RUNNING - LINE NUMBER.
C
      INCLUDE DEFINS
C
C     ..... GENERAL INFORMATION DUMP.
      WRITE(6,1) K
1     FORMAT(///' ERROR DUMP OUTPUT.  K =',I5)
  100 WRITE(6,3) M,N1,N2
3     FORMAT('   THING ',I5,' NOW BEING PROCESSED.',//,' MESSAGE ='
     1  ,1X,2A6)
C
C     ..... NODE OUTPUT.
C
 1001 MAX = MAXNO
      WRITE(6,1010) MAX,NUMNOS
 1010 FORMAT(///,' NODE OUTPUT          MAX = ',I2,12X,
     1  'NUMBER = ',I2,//,' INDEX    ID     NEXT NODE ID',
     2  '     TYPE    GEOM    INDEX LINKS IN     LINK OUT    SIGNAL',
     3  '    FSTB    CLOCK  PARAMETERS    GEN NXT',//)
      DO 101 I=1,NUMNOS
  101 WRITE(6,1011) I,NOIDNO(I),(NONXT(I,K),K=1,4),NOTYPE(I),NOGEOM(I),
     1  NOOUTP(I),NOSIG(I),FSTSG(I),NOCLK(I),
     2  (NOPARS(I,K),K=1,2),NOGNXT(I)
 1011 FORMAT(I4,6X,I2,3X,4I3,6X,I2,5X,I2,2X,16X,7X,I2,8X,I2,5X,
     1  I3,F8.0,2X,F5.0,1X,F5.0,5X,I2)
C
C     ..... SIGNAL OUTPUT
C
      MAX = MAXSIG
      LAM = AMBERT
      WRITE(6,1020) MAX,NUMSIG,LAM
 1020 FORMAT(///,' SIGNAL OUTPUT        MAX = ',I2,'       NUMBER = ',
     1  I2,'      AMBER TIME = ',I2,//,23X,'* * * *   MAJOR',
     2  '    * * * *',22X,'* * * *   MINOR    * * * *',/,
     3  ' INDEX  CYCLE',2(6X,'OFFSET  GREEN   RED    STATE CLOCK  NXT ',
     4  ))
      DO 102 I=1,NUMSIG
  102 WRITE(6,1021) I,SIGCYC(I),(SIGOFF(I,K),SIGGRN(I,K),SIGRED(I,K),
     1  SIGSTA(I,K),SIGCLK(I,K),SIGNXT(I,K),K=1,2)
 1021 FORMAT(I4,4X,I4,2(8X,I3,5X,I3,4X,I3,5X,I2,3X,I5,I4,2X))
C
C     ..... LINK OUTPUT.
C
      MAX = MAXLKS
      WRITE(6,1030) MAX,NUMLKS
 1030 FORMAT(///,' LINK OUTPUT      MAX = ',I3,'     NUMBER = ',I3,//,
     1  ' INDEX    ID NO.  LENGTH  LANES   NODE    LK OPOS    LK',
     2  ' DESTINATIONS    PROBABILITIES   ARM  LEFT  LKCAP  LKCONT  Q1',
     3  '  Q2',//)
      DO 103 I=1,NUMLKS
  103 WRITE(6,1031) I,LKID(I,1),LKID(I,2),LKLNG(I),LKLANS(I),LKNOD(I),
     1  LKOPOS(I),(LKDEST(I,K),K=1,3),(LKPROB(I,K),K=1,2),
```

```
     2  LKARM(I),LKLEFT(I),LKCAP(I),LKCONT(I),LKQ(I,1),LKQ(I,2)
 1031 FORMAT(I5,4X,2I3,I7,6X,I1,6X,I2,7X,I3,3X,2I6,3X,I3,2X,
     1  2F7.3,5X,I1,5X,I1,3X,I3,6X,I2,2X,2I4)
      IF(K .EQ. 0)  RETURN
      IF(K .EQ. 1) STOP
C
C     ..... BLOCK OUTPUT
C
      MAX = MAXBLK
      WRITE(6,1040) MAX,NBLKS
 1040 FORMAT('1BLOCK OUTPUT    MAX = ',I4,'    NUMBER = ',I4//,
     1  ' FIRST * VALUE .....',/)
      I1 = -24
  104 I1 = I1 + 25
      I2 = I1 + 24
      WRITE(6,1041) I1,(BLKARY(II),II=I1,I2)
 1041 FORMAT(I5,' *',25I4)
      IF(I2 .LT. NBLKS)  GO TO 104
C
C     ..... VEHICLE OUTPUT.
C
      MAX = MAXVEH
      WRITE(6,1050) MAX,NUMVEH
 1050 FORMAT(///,' VEHICLE OUTPUT    MAX = ',I4,'   NUMBER = ',I4//,
     1  ' INDEX   DSP   ASP   LINK   LANE   TURN   LKTIME   NEXT  TIME'
     2  ,'   STATE',//)
      DO 105 I=1,MAXVEH
      IF(VDSP(I) .EQ. 0)  GO TO 105
      WRITE(6,1051) I,VDSP(I),VLK(I),VLAN(I),VTURN(I),VLKTIM(I),
     1  VEHNXT(I),VTIME(I),VSTATE(I)
 1051 FORMAT(I4,9X,I4,3X,I4,3X,I3,5X,I3,3X,I6,3X,I4,2X,I5,4X,I2)
  105 CONTINUE
      CALL CHAIN
C
C     ..... CHAIN OUTPUT.
C
      WRITE(6,1060) VEGIND,VEGTIM,VEGFLG,MOVFST,MOVTIM,SIGIND,SIGTIM
 1060 FORMAT(///,' VEGIND = ',I5,' VEGTIM = ',I7,'  VEGFLG = ',I1,/,
     1           ' MOVFST = ',I5,' MOVTIM = ',I7,/,
     2           ' SIGIND = ',I5,' SIGTIM = ',I7)
      STOP
      END
      SUBROUTINE EXTERN(NTH)
C
C     ..... THIS SUBROUTINE READS THE DATA CARDS ON EXTERNAL
C           NODES ONLY.  CONTROL IS RETURNED TO THE CALLING
C           PROGRAM AFTER A BLANK CARD IS FOUND
C
      INCLUDE DEFINS
      INTEGER ALPHA
 1000 NTH=NTH+1
      READ(5,101) ALPHA,NOIDNO(NTH),NOTYPE(NTH),NONXT(NTH,1),
     1  NOPARS(NTH,1),NOPARS(NTH,2),NOGEOM(NTH)
      IF(ALPHA .EQ. 1H )  GO TO 2000
      GO TO 1000
 2000 NTH=NTH-1
      RETURN
```

```
  101 FORMAT(A1,I3,I2,I3,2F5.0,I1)
      END
      FUNCTION IDELAY(JTH,KTH,NTH)
C
C     ..... THIS FUNCTION CALCULATES THE AMOUNT OF DELAY.
C
      INCLUDE DEFINS
      DIMENSION INTERS(3,3)
      DATA INTERS /3,2,1,  0,0,0,  5,2,1/
      N = NOGEOM(NTH)
      K = VTURN(JTH)
      IDELAY=(CLOCK-VLKTIM(JTH)-LKLNG(KTH)*10/VDSP(JTH)-INTERS(K,N)*10)
     1                          /10
      WRITE(21) CLOCK,KTH,JTH,IDELAY
      RETURN
      END
      FUNCTION IGAP(KTH)
C
C     ..... THIS FUNCTION TESTS THE GAP THE VEHICLE IS OBSERVING.
C           A 0 MEANS GO AND A 1 MEANS STOP.
C
      INCLUDE DEFINS
      DIMENSION PROB(20),AGAP(2)
      DATA PROB / -1., -1., .02, .10, .22, .36, .50, .61, .74,
     1   .81, .84, .90, .92, .95, .96, .97, .98, .99, .995, .9995  /
C
C     ..... ARE THERE CARS ON THIS LINK TO CHECK.
C
      IF(LKCONT(KTH) .LE. 0)  GO TO 2000
C
C     ..... CHECK THE SIGNAL FIRST.
C
      NTH = LKNOD(KTH)
      ITH = NOSIG(NTH)
      I = 2 - MOD(LKARM(KTH),2)
      IF(SIGSTA(ITH,I) .NE. 0)  GO TO 2000
C
C     ..... CHECK GAP IN EACH LANE OF LINK.
C
      AGAP(1) = 1000.0
      AGAP(2) = 1000.0
      DO 300 I=1,2
      IF(LKQ(KTH,I) .EQ. 0)  GO TO 100
C
C     ..... QUEUE EXISTS, CHECK TO SEE IF LEAD CAR IS DELAYED.
      JTH = MOVEST
   10 IF(VLK(JTH) .NE. KTH)  GO TO 50
      IF(VSTATE(JTH) .GT. 3)  GO TO 50
      IF(VSTATE(JTH) .LT. 2)  GO TO 50
      IF(VLAN(JTH) .EQ. I)  GO TO 60
   50 JTH = VEHNXT(JTH)
      IF(JTH .EQ. 0)  CALL DUMP(KTH,'IGAP L','KOPOS ',33)
      GO TO 10
   60 IF(VTIME(JTH) .GE. CLOCK)  GO TO 1000
      GO TO 200
C
C     ..... NO QUEUE SO LOOK FOR THE FIRST VEHICLE.
```

```
C
  100 JTH = MOVFST
  110 IF(VLK(JTH) .NE. KTH)  GO TO 150
      IF(VSTATE(JTH) .NE. 0)  GO TO 150
      IF(VLAN(JTH) .EQ. I)  GO TO 160
  150 JTH = VEHNXT(JTH)
      IF(JTH .EQ. 0)  GO TO 200
      GO TO 110
  160 AGAP(I) = VTIME(JTH) - CLOCK
  200 IF(LKLANS(KTH) .EQ. 1)  GO TO 301
  300 CONTINUE
  301 GAP = MIN(AGAP(1),AGAP(2)) / 10.0
C
C     ..... LOOK AT SMALLEST GAP AND COMPARE TO TABLE.
C
      IF(GAP .GE. 20.0)  GO TO 2000
      IF(GAP .LE. 2)  GO TO 1000
      INDEX = GAP  + 0.5
      IF(RN1(ISEED) .LE. PROB(INDEX))  GO TO 2000
C
C     ..... REJECT.
 1000 IGAP = 1
      RETURN
C
C     ..... ACCEPT.
 2000 IGAP = 0
      RETURN
      END
      SUBROUTINE INPLKS
C
C     ..... THIS SUBROUTINE READS CARDS FOR ALL NETWORK LINKS.
C           AFTER ALL DATA CARDS HAVE BEEN READ, THE ROUTINE
C           CALCULATES SOME CROSS REFERENCE INDICES.
C
      INCLUDE DEFINS
C
C     ..... THIS IS THE LOCAL DECLARATION.
C
      INTEGER ALPHA
      KTH=0
 1000 KTH=KTH+1
      READ(5,101) ALPHA,LKID(KTH,1),LKID(KTH,2),LKLNG(KTH),LKLANS(KTH),
     1  LKOPOS(KTH),LKPROB(KTH,1),LKPROB(KTH,2),(LKDEST(KTH,K),K=1,3)
C
C     ..... IF IT IS A BLANK CARD, GO TO 2000.
      IF(ALPHA .EQ. 1H )  GO TO 2000
C
C     ..... CALCULATE THE CAPACITY OF THE LINK.
C
      LKCAP(KTH) = LKLANS(KTH)*LKLNG(KTH)/22.0
C
C     ..... FIND NODE NUMBER OF THE HEAD NODE.
C
      DO 1010 NTH=1,NUMNOS
      IF(NOIDNO(NTH) .EQ. LKID(KTH,2))  GO TO 1015
 1010 CONTINUE
      CALL DUMP(KTH,'INPLKS','DO1010',1)
```

```
 1015 LKNOD(KTH) = NTH
      DO 1016 I=1,4
      IF(LKID(KTH,1) .EQ. NONXT(NTH,I))  GO TO 1017
 1016 CONTINUE
      CALL DUMP(KTH,'INPLKS','DO1016',1)
 1017 LKARM(KTH) = I
C
C     ..... SET UP THE TURNING PROBABILITIES.
C
 1060 LKPROB(KTH,2)=LKPROB(KTH,1)+LKPROB(KTH,2)
      IF(LKPROB(KTH,2) .GE. 0.9985) LKPROB(KTH,2) = 1.0
      GO TO 1000
C
C     ..... FIND LINK OPPOSED TO LEFT TURN.
C
 2000 NUMLKS=KTH-1
      IF(NUMLKS .GE. MAXLKS)  CALL DUMP(NUMLKS,'INPLKS','MAXLKS',1)
      DO 2500 KTH=1,NUMLKS
      IF(LKOPOS(KTH) .EQ. 0) GO TO 2500
      DO 2400 KTH2=1,NUMLKS
      IF(LKID(KTH2,2) .NE. LKID(KTH,2))  GO TO 2400
      IF(LKOPOS(KTH) .EQ. LKID(KTH2,1))  GO TO 2450
 2400 CONTINUE
      CALL DUMP(KTH,'INPLKS','LKOPOS',1)
 2450 LKOPOS(KTH)=KTH2
 2500 CONTINUE
      RETURN
  101 FORMAT(A1,2I3,I4,I1,I3,2F3.3,3I3)
      END
      SUBROUTINE INTERN(NTH)
C
C     ..... THIS SUBROUTINE READS DATA CARDS ON INTERIOR
C           NODES ONLY.  CONTROL IS RETURNED TO THE CALLING
C           PROGRAM AFTER A BLANK CARD IS FOUND.
C
      INTEGER ALPHA,TEMP
      INCLUDE DEFINS
      ITH = 0
 1000 NTH = NTH + 1
      READ(5,101) ALPHA,NOIDNO(NTH),NOTYPE(NTH),(NONXT(NTH,I),I=1,4),
     1  NOGEOM(NTH)
      IF(ALPHA .EQ. 1H )  GO TO 3000
      FSTSQ(NTH) = NBLKS + 1
      NBLKS = NBLKS + 8
C
C     ..... IF NONSIGNALIZED RETURN TO READING NEXT NODE.
C
      IF(NOTYPE(NTH) .LE. 4)  GO TO 1000
C
C     ..... SIGNALIZED INTERSECTIONS REQUIRE ADDITIONAL CARD.
C
      ITH = ITH + 1
      NOSIG(NTH) = ITH
      READ(5,102) ALPHA,KDUM,SIGCYC(ITH),SIGOFF(ITH,1),SIGGRN(ITH,1),
     1  SIGOFF(ITH,2),SIGGRN(ITH,2)
C
C     ..... CALCULATE STATE CHANGE TIMES AND PUT ON CHAIN.
```

```
C
      SIGCYC(ITH) = SIGCYC(ITH) * 10
      DO 2100 I=1,2
      SIGOFF(ITH,I) = SIGOFF(ITH,I) * 10
      SIGGRN(ITH,I) = SIGGRN(ITH,I) * 10
      SIGRED(ITH,I) = SIGCYC(ITH) - SIGGRN(ITH,I) - AMBERT
      TEMP = SIGOFF(ITH,I)  + SIGGRN(ITH,I)
      IF(TEMP .GT. SIGCYC(ITH))  GO TO 2010
      TEMP = TEMP + AMBERT
      IF(TEMP .GT. SIGCYC(ITH))  GO TO 2005
      SIGSTA(ITH,I) = 2
      SIGCLK(ITH,I) = SIGOFF(ITH,I)
      GO TO 2020
 2005 SIGSTA(ITH,I) = 1
      SIGCLK(ITH,I) = TEMP - SIGCYC(ITH)
      GO TO 2020
 2010 SIGSTA(ITH,I) = 0
      SIGCLK(ITH,I) = TEMP - SIGCYC(ITH)
C
C     ..... PUT SIGNAL ON EVENTS CHAIN.  FIRST CHECK EMPTY CHAIN.
C
 2020 IF(ITH*I .EQ. 1)  GO TO 2090
C
C     ..... NEXT SEE IF IT SHOULD BE FIRST.
C
      ITHOLD = ABS(SIGIND)
      IOLD = 1
      IF(SIGIND .LE. 0) IOLD = 2
      IF(SIGCLK(ITH,I) .LE. SIGCLK(ITHOLD,IOLD))  GO TO 2080
C
C     ..... MUST BRACKET BETWEEN AN OLD AND A NEW.
C
 2025 ITHNEW = SIGNXT(ITHOLD,IOLD)
      IF(ITHNEW) 2050,2030,2035
 2030 INEW = 1
      GO TO 2075
 2035 INEW = 1
      GO TO 2055
 2050 ITHNEW = -ITHNEW
      INEW = 2
C
C     ..... IF NEW BIGGER, HAVE FOUND SPOT TO INSERT.
C
 2055 IF(SIGCLK(ITHNEW,INEW) .GE. SIGCLK(ITH,I))  GO TO 2075
C
C     ..... OTHERWISE CHANGE NEW TO OLD AND CHECK NEXT SLOT.
C
      ITHOLD = ITHNEW
      IOLD = INEW
      GO TO 2025
C
C     ..... POSITION FOUND PUT ON CHAIN.
C
 2075 IF(I .EQ. 1)  GO TO 2076
      SIGNXT(ITHOLD,IOLD) = -ITH
      GO TO 2077
 2076 SIGNXT(ITHOLD,IOLD) = ITH
```

```
 2077 IF(INEW .EQ. 1)  GO TO 2078
      SIGNXT(ITH,I) = -ITHNEW
      GO TO 2100
 2078 SIGNXT(ITH,I) = ITHNEW
      GO TO 2100
C
C     ..... PUT AT HEAD OF CHAIN.
C
 2080 SIGNXT(ITH,I) = SIGIND
      SIGIND = ITH
      IF(I .EQ. 2) SIGIND = -ITH
      SIGTIM = SIGCLK(ITH,I)
      GO TO 2100
C
C     ..... FIRST SIGNAL ON CHAIN.
C
 2090 SIGIND = 1
      SIGTIM = SIGCLK(1,1)
C
C     ..... END OF DO LOOP.
 2100 CONTINUE
C
C     ..... GO BACK AND READ DATA ON NEXT NODE.
C
      GO TO 1000
C
C     ..... SET PARAMETERS.
C
 3000 NUMNOS = NTH - 1
      IF(NUMNOS .GE. MAXNO)  CALL DUMP(NUMNOS,'INTERN','MAXNO ',1)
      IF(NBLKS .GT. MAXBLK)  CALL DUMP(NBLKS,'INTERN','MAXBLK',1)
      NUMSIG = ITH
      IF(NUMSIG .GE. MAXSIG)  CALL DUMP(NUMSIG,'INTERN','MAXSIG',1)
      RETURN
C
C     ..... FORMAT STATEMENTS
C
  101 FORMAT(A1,I3,I2,4I3,I1)
  102 FORMAT(A1,I3,5I3)
      END
      SUBROUTINE ISQ(I,KTH,NTH,JTH,IARM,I1,I2)
C
C     ..... THIS SUBROUTINE CHECKS THE SQUARE INDICATED BY I.
C           IF A SQUARE IS AVAILABLE, THE OUTPUT IS THE BLOCK
C           INDEX.  IF UNAVAILABLE, THE OUTPUT IS THE NEGATIVE
C           OF THE VEHICLE INDEX.
C
      INCLUDE DEFINS
      DIMENSION IADD(4,4)
      DATA IADD /3,4,2,1,4,2,1,3,2,1,3,4,1,3,4,2/
      IF(NOGEOM(NTH) .NE. 1)  GO TO 200
C
C     ..... TWO-BY-TWO.
C
  100 I2 = 0
      IB = FSTSQ(NTH) + (IADD(IARM,I) - 1) * 2
      IF(BLKARY(IB) .NE. 0)  GO TO 110
```

```
      I1 = IB
      RETURN
  110 I1 = -BLKARY(IB)
      RETURN
C
C     ..... FOUR BY FOUR.
C
  200 CONTINUE
      IB = FSTSQ(NTH) + (IADD(IARM,I) - 1) * 2
      IF(BLKARY(IB) .NE. 0)  GO TO 210
      I1 = IB
      GO TO 230
  210 I1 = -BLKARY(IB)
  230 IB = IB + 1
      IF(BLKARY(IB) .NE. 0)  GO TO 240
      I2 = IB
      RETURN
  240 I2 = -BLKARY(IB)
      RETURN
      END
      SUBROUTINE ISQOUT(I,KTH,NTH,JTH,IARM)
C
C     ..... THIS SUBROUTINE FINDS THE BLOCK THE VEHICLE WAS IN
C           AND REMOVES THE VEHICLE.
C
      INCLUDE DEFINS
      DIMENSION IADD(4,4)
      DATA IADD/3,4,2,1,4,2,1,3,2,1,3,4,1,3,4,2/
C
      IB = FSTSQ(NTH) + (IADD(IARM,I)-1)*2
      IF(BLKARY(IB) .NE. JTH)  GO TO 15
    5 BLKARY(IB) = 0
      IF(I .NE. 3)  RETURN
   10 IB = IB + 1
      BLKARY(IB) = 0
      RETURN
   15 IB = IB + 1
      IF(BLKARY(IB) .NE. JTH)  CALL DUMP(JTH,'ISQOUT','NOFIND',18)
      BLKARY(IB) = 0
      RETURN
      END
      SUBROUTINE MOVVEH
C
C     ..... THIS SUBROUTINE MOVES VEHICLES ON THE VEHICLE MOVE
C           CHAIN.  THE ORDER IN WHICH VEHICLES ARE PROCESSED
C           DEPENDS ON THEIR STATE AND  IS SPECIFIED BY THE
C           ARRAY CALLED ORDER.
C
      INCLUDE DEFINS
      PARAMETER FSTGAP = 38
      INTEGER DELFAC,DISGAP,TIME,ORDER(7),DELAY
      DIMENSION ITIME(3,3,3),DISGAP(5)
      DEFINE TIME(I,J,K) = ITIME(I,J,K)
      DATA DISGAP / 31, 27, 24, 22, 21 /
      DATA ITIME /1,1,1, 1,1,0, 1,0,0, 0,0,0, 0,0,0, 0,0,0,
     1  2,1,2, 1,1,0, 1,0,0/
      DATA ORDER/0,6,5,4,2,3,1/
```

```
C**** WRITE(6,1)
    1 FORMAT(' MOVVEH SUBROUTINE.')
C**** CALL CHAIN
      DO 1000 M=1,6
      INDEX = ORDER(M)
C
C     ..... SET UP TO START AT HEAD OF CHAIN.
C
      KGOTO = INDEX + 1
   25 JTHLST = 0
      JTH = MOVFST
      IF(JTH .EQ. 0)  GO TO 1010
C
C     ..... CHECK EVENT TIME OF THIS VEHICLE.
C
   50 IF(VTIME(JTH) .GT. CLOCK)  GO TO 1000
      JTHNXT = VEHNXT(JTH)
C
C     ..... IS THIS VEHICLE IN THE RIGHT STATE.
C
      IF(VSTATE(JTH) .NE. INDEX)  GO TO 975
      KTH = VLK(JTH)
      NTH = LKNOD(KTH)
      ILANE = VLAN(JTH)
C
C     ..... IF THIS IS A TERMINATE NODE, SKIP OUT OF THIS AREA.
C
      IF(NOTYPE(NTH) .LT. 3)  GO TO 800
C
C     ..... BRANCH BASED ON THE STATE OF THE VEHICLE.
C
      GO TO (100,975,300,400,500,600,700),KGOTO
  100 CONTINUE
C
C     ..... AN ARRIVAL, FIRST CHECK THE QUEUE.
C
      DELFAC = 0
      IF(LKQ(KTH,ILANE) .EQ. 0)  GO TO 112
C
C     ..... PUT IN QUEUE.  NO CHANGE IN CHAIN POSITION.
C
      LKQ(KTH,ILANE) = LKQ(KTH,ILANE) + 1
      IF(LKQ(KTH,ILANE) .GT. LKMAXQ(KTH,ILANE))  LKMAXQ(KTH,ILANE) =
     1              LKQ(KTH,ILANE)
      LKSTOP(KTH,ILANE) = LKSTOP(KTH,ILANE) + 1
      VSTATE(JTH) = 1
C**** WRITE(6,3)  JTH,KTH,ILANE,VSTATE(JTH)
      GO TO 975
C
C     ..... BRANCH BASED ON LIGHT STATE.
C
  112 ITH = NOSIG(NTH)
      I2 = 2 - MOD(LKARM(KTH),2)
      IF(SIGSTA(ITH,I2) - 1)  160,120,115
C
C     ..... RED LIGHT, RESCHEDULE MOVE EVENT.
C
```

```
  115 VTIME(JTH) = SIGCLK(ITH,I2) + FSTGAP
      VTIME(JTH) = VTIME(JTH) / 10 * 10
      GO TO 122
C
C     ..... AMBER LIGHT, CHECK FOR A LEFT TURN.
C
  120 IF(VTURN(JTH) .EQ. 1)  GO TO 125
C
C     ..... STRAIGHT AND RIGHT TURNS STOP FOR LIGHT.
C
  121 VTIME(JTH) = FSTGAP + SIGCLK(ITH,I2) + SIGRED(ITH,I2)
      VTIME(JTH) = VTIME(JTH) / 10 * 10
  122 IF(DELFAC .GT. 0)  GO TO 123
      LKQ(KTH,ILANE) = 1
      IF(LKMAXQ(KTH,ILANE) .EQ. 0)  LKMAXQ(KTH,ILANE) = 1
      LKSTOP(KTH,ILANE) = LKSTOP(KTH,ILANE) + 1
  123 VSTATE(JTH) = 2
      LKDIS(KTH,ILANE) = 0
C**** WRITE(6,6) JTH,KTH,ILANE,VTIME(JTH)
      GO TO 950
C
C     ..... LEFT TURNS WHO WERE BLOCKED WILL TRY TO SNEAK THROUGH.
C
  125 IF(INDEX .NE. 3)  GO TO 121
      KTHNXT = LKDEST(KTH,1)
      IF(LKCAP(KTHNXT) .LE. LKCONT(KTHNXT))  GO TO 121
C
C     ..... BRANCH ON NUMBER OF LANES.
      IF(LKLANS(KTH) .EQ. 2)  GO TO 127
C
C     ..... SINGLE LANE MUST HAVE S1 OPEN AND NO LEFT TURNS
C           IN S2.
C
      CALL ISQ(1,KTH,NTH,JTH,LKARM(KTH),IB,IB2)
      IF(IB .LT. 0)  GO TO 121
      CALL ISQ(2,KTH,NTH,JTH,LKARM(KTH),IB2,IB1)
      GO TO 130
C
C     ..... TWO LANE STREET REQUIRES SPACE IN S1 AND NO LEFT
C           TURN IN EITHER S1 OR S2.
C
  127 CALL ISQ(1,KTH,NTH,JTH,LKARM(KTH),IB1,IB2)
      IF(IB1 .LT. 0 .AND. IB2 .LT. 0)  GO TO 121
      IB = MIN(IB1,IB2)
      IF(IB .GT. 0)  GO TO 128
      JTHCHK = -IB
      IF(VTURN(JTHCHK) .EQ. 1)  GO TO 121
      IB = MAX(IB1,IB2)
C
  128 CALL ISQ(2,KTH,NTH,JTH,LKARM(KTH),IB1,IB2)
      IF(IB1 .GT. 0)  GO TO 130
      IF(VTURN(-IB1) .EQ. 1)  GO TO 121
  130 IF(IB2 .GT. 0)  GO TO 135
      IF(VTURN(-IB2) .EQ. 1)  GO TO 121
C
C     ..... MOVE THE VEHICLE INTO THE INTERSECTION.
C
```

```
  135 VTIME(JTH) = CLOCK + TIME(1,1,NOGEOM(NTH)) * 10 + DELFAC
      VTIME(JTH) = VTIME(JTH) / 10 * 10
      GO TO 940
C
C     ..... GREEN SIGNAL, FIRST CHECK S1 FOR SPACE.
C
  160 CALL ISQ(1,KTH,NTH,JTH,LKARM(KTH),IB1,IB2)
      IF(IB1 .GT. 0 .OR. IB2 .GT. 0)  GO TO 165
C
C     ..... VEHICLE IS BLOCKED.  PUT IN STATE 3 AND HOLD THE
C           PRESENT CHAIN POSITION.
C
  162 VSTATE(JTH) = 3
C**** WRITE(6,7) JTH,KTH,ILANE
      IF(DELFAC .GT. 0)  GO TO 975
      LKQ(KTH,ILANE) = 1
      IF(LKMAXQ(KTH,ILANE) .EQ. 0)  LKMAXQ(KTH,ILANE) = 1
      LKSTOP(KTH,ILANE) = LKSTOP(KTH,ILANE) + 1
  163 GO TO 975
C
C     ..... NEXT CHECK CAPACITY OF EXIT LINK AND PUT IN
C           THE BLOCKED STATE IF LINK IS FULL.
C
  165 K = VTURN(JTH)
      KTHNXT = LKDEST(KTH,K)
      IF(LKCAP(KTHNXT) .LE. LKCONT(KTHNXT))  GO TO 162
C
C     ..... OTHERWISE BRANCH ON TURNING MOVEMENT.
C
      GO TO (190,175,170),K
C
C     ..... RIGHT TURN TRAFFIC.
C
  170 IB = MAX(IB1,IB2)
      VTIME(JTH) = TIME(1,3,NOGEOM(NTH)) * 10 + DELFAC + CLOCK
      VTIME(JTH) = VTIME(JTH) / 10 * 10
      GO TO 940
C
C     ..... STRAIGHT TRAFFIC.
C
  175 IB = MAX(IB1,IB2)
      IF(LKLANS(KTH) .EQ. 2)  GO TO 180
      CALL ISQ(2,KTH,NTH,JTH,LKARM(KTH),IB1,IB2)
      IF(IB1 .GT. 0)  GO TO 180
      IF(VTURN(-IB1) .EQ. 1)  GO TO 162
  180 VTIME(JTH) = TIME(1,2,NOGEOM(NTH)) * 10 + DELFAC + CLOCK
      VTIME(JTH) = VTIME(JTH) / 10 * 10
      GO TO 940
C
C     ..... LEFT TURN VEHICLES.  CHECK FOR LEFT TURN IN FRONT.
C
  190 IB = MIN(IB1,IB2)
      IF(IB .GE. 0)  GO TO 192
      IF(VTURN(-IB) .EQ. 1)  GO TO 162
  192 IB = MAX(IB1,IB2)
      CALL ISQ(2,KTH,NTH,JTH,LKARM(KTH),IB1,IB2)
      IF(IB1 .GT. 0)  GO TO 195
```

Page missing from thesis

```
      CALL ISQOUT(2,KTH,NTH,JTH,LKARM(KTH))
      DELFAC = 0
      IF(VTIME(JTH) .NE. CLOCK)  DELFAC = 10
      GO TO 900
C
C     ..... LEFT TURN MUST CHECK GAP AND S3
  610 IF(IGAP(LKOPOS(KTH)) .EQ. 1)  GO TO 975
      CALL ISQ(3,KTH,NTH,JTH,LKARM(KTH),IB1,IB2)
      IF(IB1 .LT. 0 .OR.  IB2 .LT. 0)  GO TO 975
      BLKARY(IB1) = JTH
      IF(IB2 .GT. 0)  BLKARY(IB2) = JTH
C**** WRITE(6,10) JTH,NTH
      CALL ISQOUT(2,KTH,NTH,JTH,LKARM(KTH))
      VSTATE(JTH) = 6
      DELFAC = 0
      IF(VTIME(JTH) .NE. CLOCK)  DELFAC = 10
      VTIME(JTH)=TIME(3,1,NOGEOM(NTH)) * 10 + DELFAC + CLOCK
      VTIME(JTH) = VTIME(JTH) / 10 * 10
      GO TO 950
C
C     ..... VEHICLE TO EXIT S3.  CHECK CAPACITY.
C
  700 KTHNXT = LKDEST(KTH,1)
      IF(LKCAP(KTHNXT) .LE. LKCONT(KTHNXT))  GO TO 975
      DELFAC = 0
      CALL ISQOUT(3,KTH,NTH,JTH,LKARM(KTH))
      GO TO 900
C
C     ..... TERMINATION SECTION.
C
  800 NUMVEH = NUMVEH - 1
      ITURN = VTURN(JTH)
      LKVL(KTH,ILANE) = LKVL(KTH,ILANE) + 1
      LKVD(KTH,ITURN) = LKVD(KTH,ITURN) + 1
      VDSP(JTH) = 0
      VEHNXT(JTH) = 0
      LKCONT(KTH) = LKCONT(KTH) - 1
C**** WRITE(6,2) JTH,NTH
      IF(JTHLST .EQ. 0)  GO TO 810
      VEHNXT(JTHLST) = JTHNXT
      GO TO 980
  810 MOVFST = JTHNXT
      GO TO 980
C
C     ..... ADVANCE VEHICLE TO NEW LINK.
C
  900 DELAY = IDELAY(JTH,KTH,NTH) + DELFAC
      ITURN = VTURN(JTH)
      LKVL(KTH,ILANE) = LKVL(KTH,ILANE) + 1
      LKDEL(KTH,ILANE) = LKDEL(KTH,ILANE) + DELAY
      LKVD(KTH,ITURN) = LKVD(KTH,ITURN) + 1
      LKDELD(KTH,ITURN) = LKDELD(KTH,ITURN) + DELAY
      LKCONT(KTHNXT) = LKCONT(KTHNXT) + 1
      VSTATE(JTH) = 0
C**** WRITE(6,8) JTH,KTHNXT
      VLK(JTH) = KTHNXT
      X = RN1(ISEED)
```

```
      DO 905 I=1,2
      IF(X .LE. LKPROB(KTHNXT,I))  GO TO 910
  905 CONTINUE
      VTURN(JTH) = 3
      VLAN(JTH) = LKLANS(KTHNXT)
      GO TO 920
  910 VTURN(JTH) = I
      IF(I .EQ. 2)  GO TO 915
      VLAN(JTH) = 1
      GO TO 920
  915 VLAN(JTH) = RN1(ISEED) * LKLANS(KTHNXT) + 1
  920 VLKTIM(JTH) = CLOCK + DELFAC
      VTIME(JTH) = LKLNG(KTHNXT) * 10 / VDSP(JTH) + DELFAC + CLOCK
      ILANE = VLAN(JTH)
      IF(VTIME(JTH) .LT. LKARVT(KTHNXT,ILANE ) + 10) VTIME(JTH) =
     1   LKARVT(KTHNXT,ILANE) + 10
      VTIME(JTH) = VTIME(JTH) / 10 * 10
      LKARVT(KTHNXT,ILANE) = VTIME(JTH)
      IF(JTHLST .NE. 0)  GO TO 925
      MOVFST = JTHNXT
      GO TO 930
  925 VEHNXT(JTHLST) = JTHNXT
  930 CALL VCHAIN(JTH,VTIME(JTH))
      IF(JTHNXT .EQ. VEHNXT(JTH)) JTHLST = JTH
      GO TO 980
C
C     ..... MOVE INTO S1.
C
  940 BLKARY(IB) = JTH
      VSTATE(JTH) = 4
      LKQCLK(KTH,ILANE) = CLOCK
C**** WRITE(6,5) JTH,KTH,ILANE,VTIME(JTH)
C
C     ..... FIND NEW POSITION ON CHAIN.
C
  950 IF(JTHLST .NE. 0)  GO TO 955
      MOVFST = JTHNXT
      GO TO 960
  955 VEHNXT(JTHLST) = JTHNXT
  960 CALL VCHAIN(JTH,VTIME(JTH))
      IF(JTHNXT .EQ. VEHNXT(JTH)) JTHLST = JTH
      IF(VSTATE(JTH) .NE. 4)  GO TO 980
C
C     ..... VEHICLES MOVED TO S1 REQUIRE ADDITIONAL PROCESSING.
C
      LKCONT(KTH) = LKCONT(KTH) - 1
      IF(DELFAC .NE. 0)  LKQ(KTH,ILANE) = LKQ(KTH,ILANE) - 1
      IF(LKQ(KTH,ILANE) .GT. 0)  GO TO 961
      LKDIS(KTH,ILANE) = 0
      GO TO 980
  961 N = LKDIS(KTH,ILANE) + 1
      LKDIS(KTH,ILANE) = N
      IF(N .GT. 5)  N = 5
      JTHLSP = 0
      JTHP = MOVFST
  962 JTHNXP = VEHNXT(JTHP)
      IF(VTIME(JTHP) .GT. CLOCK)  CALL DUMP(KTH,'MOVVEH','QERCH',342)
```

```
      IF(VLK(JTHP) .NE. KTH)  GO TO 963
      IF(VSTATE(JTHP) .NE. 1)  GO TO 963
      IF(VLAN(JTHP) .EQ. ILANE)  GO TO 965
  963 JTHLSP = JTHP
      JTHP = JTHNXP
      IF(JTHP .EQ. 0)  CALL DUMP(KTH,'MOVVEH','QSERCH',348)
      GO TO 962
C
C     ..... NEW LEADER IN QUEUE, PULL OUT AND SCHEDULE TO DISCHARGE.
C
  965 IF(JTHLSP .EQ. 0)  GO TO 967
      VEHNXT(JTHLSP) = JTHNXP
      GO TO 968
  967 MOVFST = JTHNXP
  968 VTIME(JTHP) = (CLOCK + DISGAP(N)) / 10 * 10
      VSTATE(JTHP) = 3
      CALL VCHAIN(JTHP,VTIME(JTHP))
C**** WRITE(6,4) JTHP,KTH,ILANE,VTIME(JTHP)
      GO TO 25
C
C     ..... NO CHANGE IN CHAIN POSITION.
C
  975 JTHLST = JTH
C
C     ..... CHECK NEXT VEHICLE.
C
  980 IF(JTHNXT .EQ. 0)  GO TO 1000
      JTH = JTHNXT
      GO TO 50
C
C     ..... END OF DO LOOP.
C
 1000 CONTINUE
C
C     ..... FIND TIME OF NEXT EVENT
C
      JTH = MOVFST
 1001 IF(JTH .EQ. 0)  GO TO 1010
      IF(VTIME(JTH) .GT. CLOCK)  GO TO 1005
      JTH =VEHNXT(JTH)
      GO TO 1001
 1005 MOVTIM = VTIME(JTH)
      RETURN
 1010 MOVTIM = 2**30
      RETURN
    2 FORMAT('   TERMINATE ',I4,' AT NODE ',I4)
    3 FORMAT('   VEH ',I4,' PUT IN QUEUE ',I4,I2,' STATE ',I2)
    4 FORMAT('       ',I4,' TO EXIT QUEUE ',I4,I2,' AT TIME ',I7)
    5 FORMAT('   VEH ',I4,' EXIT QUEUE ',I4,I2,' TO S1 WILL MOVE',I7)
    6 FORMAT('   VEH ',I4,' LIGHT STOP QUEUE ',I4,I2,' WILL MOVE',I7)
    7 FORMAT('   VEH ',I4,' BLOCKED AT QUEUE ',I4,I2)
    8 FORMAT('   VEH ',I4,' ADVANVED TO NEW LINK ',I4)
    9 FORMAT('   VEH ',I4,' EXIT S1 MOVES TO S2 AT NODE ',I4)
   10 FORMAT('   VEH ',I4,' EXIT S2 MOVES TO S3 AT NODE ',I4)
      END
      FUNCTION RN1(ISEED)
C
```

```
C      ..... THIS FUNCTION GENERATES UNIFORMLY DISTRIBUTED
C            RANDOM NUMBERS BETWEEN 0 AND 1
C
      ISEED=ISEED*185333
      IF(ISEED .LT. 0) ISEED=ISEED+34359738367+1
      RN1=ISEED*2.0**(-35)
      RETURN
      END
      SUBROUTINE SETUP
C
C      ..... THIS SUBROUTINE DOES THE FINAL SETUP ON ALL DATA
C            FILES BEFORE THE SIMULATION BEGINS.
C
      INCLUDE DEFINS
      VEGTIM = 2**30
      VEGIND = 0
C
C      ..... FOR EACH NODE
      DO 2000 NTH=1,NUMNOS
      IF(NOTYPE(NTH) .GT. 2) GO TO 1500
      IF(NOTYPE(NTH) .EQ. 2) GO TO 2000
C
C      ..... GENERATE NODES, SET TIME OF FIRST ARRIVAL.
      NOPARS(NTH,1) = 36000. / NOPARS(NTH,1)
      DO 1001 I=1,100
 1001 A = RN1(ISEED)
      NOSEED(NTH) = ISEED
      NOCLK(NTH) = TBAGEN(NTH)
C
C      ..... FIND LINK TO PUT VEHICLES ON.
      DO 1100 KTH=1,NUMLKS
      IF(NONXT(NTH,1) .EQ. LKID(KTH,2) .AND. NOIDNO(NTH) .EQ. LKID(KTH,1
     1))    GO TO 1110
 1100 CONTINUE
      CALL DUMP(NTH,'SETUP ','DO1100',1)
 1110 NOOUTP(NTH) = KTH
C
C      ..... PUT ON CHAIN.  IF CLOCK IS SMALL PUT FIRST.
C
      IF(NOCLK(NTH) .GT. VEGTIM)  GO TO 1210
      VEGTIM = NOCLK(NTH)
      NOGNXT(NTH) = VEGIND
      VEGIND = NTH
      GO TO 2000
C
C      ..... PUT ON CHAIN BETWEEN TWO NODES.
C
 1210 NTHLSP = VEGIND
 1220 NTHP = NOGNXT(NTHLSP)
      IF(NTHP .EQ. 0)  GO TO 1240
      IF(NOCLK(NTH) .LE. NOCLK(NTHP))  GO TO 1240
      NTHLSP = NTHP
      GO TO 1220
 1240 NOGNXT(NTH) = NTHP
      NOGNXT(NTHLSP) = NTH
      GO TO 2000
C
```

```
C      ..... INTERNAL NODES
C
 1500 CONTINUE
      ITH = NOSIG(NTH)
      IF(ITH .EQ. 0)  GO TO 2000
      IDNTH = NOIDNO(NTH)
C
C      ..... TAKE EACH LINK AND SEE IF IT IS CONNECTED TO THIS
C            NODE.
C
      DO 1600 KTH=1,NUMLKS
C
C      ..... CHECK FOR INPUT TO THIS NODE.
      IF(LKID(KTH,2) .NE. IDNTH) GO TO 1600
C
C      ..... OK, NOW FIND OUT WHERE IT CAME FROM.
      DO 1520 I=1,4
      IF(LKID(KTH,1) .EQ. NONXT(NTH,I))  GO TO 1525
 1520 CONTINUE
      CALL DUMP(KTH,'SETUP ','D0152O',1)
 1525 SIGINP(ITH,I) = KTH
 1600 CONTINUE
 2000 CONTINUE
C
C      ..... FOR EACH LINK FIND THE DESTINATIONS.
      DO 3000 KTH=1,NUMLKS
      DO 2500 I=1,3
      IF(LKDEST(KTH,I) .EQ. 0)  GO TO 2500
      DO 2400 KTHCHK=1,NUMLKS
      IF(LKID(KTHCHK,2) .NE. LKDEST(KTH,I))  GO TO 2400
      IF(LKID(KTHCHK,1) .NE. LKID(KTH,2))  GO TO 2400
      LKDEST(KTH,I) = KTHCHK
      GO TO 2500
 2400 CONTINUE
 2500 CONTINUE
 3000 CONTINUE
      RETURN
      END
      SUBROUTINE SIGCHK
C
C      ..... THIS SUBROUTINE UPDATES A SIGNAL LIGHT.  ADDITIONALLY,
C            THE OTHER LIGHTS ARE CHECKED FOR A STATE CHANGE AT
C            THE CURRENT CLOCK TIME.
C
      INCLUDE DEFINS
C
C      ..... FIRST ESTABLISH MAJOR OR MINOR AND IF CHANGE IS REALLY
C            WANTED NOW.
C
   10 ITH = ABS(SIGIND)
      I = 1
      IF(SIGIND .LE. 0)  I = 2
      IF(SIGCLK(ITH,I) .GT. CLOCK)  GO TO 700
C
C      ..... BRANCH BASED ON OLD STATE.
C
      IF(SIGSTA(ITH,I)-1)  100,200,300
```

```
C
C     ..... OLD STATE WAS GREEN, CHANGE TO AMBER.
C
  100 SIGSTA(ITH,I) = 1
      SIGCLK(ITH,I) = SIGCLK(ITH,I) + AMBERT
      GO TO 500
C
C     ..... OLD STATE WAS AMBER, CHANGE TO RED.
C
  200 SIGSTA(ITH,I) = 2
      SIGCLK(ITH,I) = SIGCLK(ITH,I) + SIGRED(ITH,I)
      GO TO 500
C
C     ..... OLD STATE WAS RED, CHANGE TO GREEN.
C
  300 SIGSTA(ITH,I) = 0
      SIGCLK(ITH,I) = SIGCLK(ITH,I) + SIGGRN(ITH,I)
C
C     ..... FIND THE NEXT SIGNAL TO CHANGE AND THEN PUT THIS
C           SIGNAL ON THE CHAIN.
C
  500 SIGIND = SIGNXT(ITH,I)
C
      ITHOLD =ABS(SIGIND)
      IOLD = 1
      IF(SIGIND .LE. 0)  IOLD = 2
C
      IF(SIGCLK(ITH,I) .GT. SIGCLK(ITHOLD,IOLD))  GO TO 525
C
C     .....PUT BACK AT HEAD OF CHAIN.
C
      SIGIND = ITH
      IF(I .EQ. 2)  SIGIND = -ITH
      GO TO 700
C
C     ..... BRACKET A POSITION ON THE CHAIN - BETWEEN AN
C           OLD AND A NEW.
C
  525 ITHNEW = SIGNXT(ITHOLD,IOLD)
      IF(ITHNEW)  550,530,535
  530 INEW = 1
      GO TO 575
  535 INEW = 1
      GO TO 555
  550 ITHNEW = -ITHNEW
      INEW = 2
C
C     ..... IF TIME FOR NEW BIGGER, HAVE FOUND THE SPOT
C
  555 IF(SIGCLK(ITHNEW,INEW) .GE. SIGCLK(ITH,I))  GO TO 575
C
C     ..... CHANGE NEW TO OLD AND CHECK NEXT SPOT.
C
      ITHOLD = ITHNEW
      IOLD = INEW
      GO TO 525
C
```

```
C     ..... POSITION FOUND, PUT SIGNAL ON THE CHAIN.
C
  575 IF(I .EQ. 1)  GO TO 576
      SIGNXT(ITHOLD,IOLD) = -ITH
      GO TO 580
  576 SIGNXT(ITHOLD,IOLD) = ITH
  580 IF(INEW .EQ. 1)  GO TO 581
      SIGNXT(ITH,I) = -ITHNEW
      GO TO 10
  581 SIGNXT(ITH,I) = ITHNEW
      GO TO 10
C
C     ..... LAST SIGNAL TO PROCESS NOW.
C
  700 SIGTIM = SIGCLK(ITH,I)
C**** WRITE(6,1) SIGTIM,ITH,I,((SIGCLK(K,J),SIGSTA(K,J),J=1,2),K=1,
C****1  NUMSIG)
    1 FORMAT(' SIGCHK SUBROUTINE  SIGTIM =',I9,'  ITH = ',
     1  I5,'I = ',I2,/,' CLK,STA: ',10(I9,',',I1),/,8X,10(I9,',',I1))
          RETURN
      END
      FUNCTION SPDDIS(ISEED)
      DIMENSION ARY(4)
      DATA ARY / -1., 0.07, 0.97, 1.0 /
      X=RN1(ISEED)
      DO 100 I=2,4
      IF(ARY(I) .GE. X)  GO TO 200
  100 CONTINUE
  200 SPDDIS = I * 18.0
      RETURN
      END
      SUBROUTINE STAT(K)
C
C     ..... THIS SUBROUTINE CLEARS THE STATISTICS AFTER WARMUP
C           AND THEN PRINTS THEM AFTER THE FINISH.
C
      INCLUDE DEFINS
C
C     ..... IF FINISH, GO TO 1000
      IF(K .EQ. 2)  GO TO 1000
      DO 200 KTH = 1,NUMLKS
      DO 100 I=1,2
      LKMAXQ(KTH,I) = LKQ(KTH,I)
      LKSTOP(KTH,I) = 0
      LKDEL(KTH,I) = 0
  100 LKVL(KTH,I) = 0
      DO 200 I=1,3
      LKDELD(KTH,I) = 0
  200 LKVD(KTH,I) = 0
      RETURN
C
C     ..... PRINT THE STATISTICS IN THIS SECTION.
C
 1000 TOTIM = FINISH - WARMUP
      WRITE(6,1)
    1 FORMAT(1H1,21X,'VEHICLE COUNT',15X,'VEHICLE VOLUME',21X,
     1  'VEHICLE DELAY',14X,'DELAY PER VEHICLE',/,' LINK   LANES   ',
```

```
     2  'LANE 1   LANE 2   TOTAL      LANE 1   LANE 2   TOTAL',
     3  12X,'LANE 1   LANE 2   TOTAL        LANE 1   LANE 2   TOTAL',/)
      DO 1100 KTH=1,NUMLKS
      LKVTOT = LKVL(KTH,1) + LKVL(KTH,2)
      LKVOL1 = LKVL(KTH,1) * 36000 / TOTIM
      LKVOL2 = LKVL(KTH,2) * 36000 / TOTIM
      LKVOLT = LKVOL1 + LKVOL2
      LKDELT = LKDEL(KTH,1) + LKDEL(KTH,2)
      DELVL1 = LKDEL(KTH,1) / FLOAT(LKVL(KTH,1))
      DELVL2 = LKDEL(KTH,2) / FLOAT(LKVL(KTH,2))
      DELVLT = LKDELT / FLOAT(LKVL(KTH,1) + LKVL(KTH,2))
1100  WRITE(6,2) KTH,LKLANS(KTH),LKVL(KTH,1),LKVL(KTH,2),LKVTOT,
     1  LKVOL1,LKVOL2,LKVOLT,LKDEL(KTH,1),LKDEL(KTH,2),LKDELT,
     2  DELVL1,DELVL2,DELVLT
    2 FORMAT(I4,4X,I2,2(1X,3I9),9X,3I9,2X,3F9.2)
      WRITE(6,3)
    3 FORMAT(///,14X,'VEHICLE STOPS',11X,'MAX QUEUES',14X,'TURNS',
     1  19X,'TURN DELAY',14X,'TURN DELAY PER VEHICLE',/,
     2  ' LINK   LANE 1   LANE 2   TOTAL    LANE 1   LANE 2    LEFT',
     3  '  STRAIGHT RIGHT      LEFT    STRAIGHT  RIGHT       LEFT',
     4  '   STRAIGHT  RIGHT',/)
      DO 1200 KTH=1,NUMLKS
      LKSTOT = LKSTOP(KTH,1) + LKSTOP(KTH,2)
      DELVL1 = LKDELD(KTH,1) / LKVD(KTH,1)
      DELVL2 = LKDELD(KTH,2) / LKVD(KTH,2)
      DELVL3 = LKDELD(KTH,3) / LKVD(KTH,3)
1200  WRITE(6,4) KTH,LKSTOP(KTH,1),LKSTOP(KTH,2),LKSTOT,LKMAXQ(KTH,1),
     1  LKMAXQ(KTH,2),(LKVD(KTH,I),I=1,3),(LKDELD(KTH,I),I=1,3),
     2  DELVL1,DELVL2,DELVL3
    4 FORMAT(I4,3(4X,I5),6X,I2,7X,I2,2X,3(3X,I5),2(4X,I6),3X,I6,
     1  2X,3(1X,F8.2))
    5 FORMAT(//,23H INTERSECTION ID NUMBER,I4,/,12HOINPUT LINKS,6X,
     1    14HVEHICLE VOLUME,10X,9HMAX QUEUE,11X,13HVEHICLE DELAY,12X,
     2    17HDELAY PER VEHICLE,/,14X,22HLANE 1  LANE 2   TOTAL,4X,
     3    14HLANE 1  LANE 2,2(4X,22HLANE 1  LANE 2   TOTAL),/)
    6 FORMAT(I7,4X,2I8,I9,2I8,3X,2I8,I9,2X,2F8.2,F9.2)
    7 FORMAT(30X,6H------,64X,6H------,/,21H  TOTAL INTERSECTION ,
     1    8HVOLUME = ,I7,23X,40HAVERAGE INTERSECTION DELAY PER VEHICLE =,
     2    F7.2)
    8 FORMAT(/////,32H AVERAGE LINK DELAY FOR SYSTEM = F7.2)
    9 FORMAT(29H1INTERSECTION OUTPUT SUMMARY.)
      WRITE(6,9)
      NTOT = 0
      NDTOT = 0
      DO 1275 NTH=1,NUMNOS
      IF(NOTYPE(NTH) .LT. 3)  GO TO 1275
      NINT = 0
      NDEL = 0
      WRITE(6,5) NOIDNO(NTH)
      ITH = NOSIG(NTH)
      DO 1250 II=1,4
      KTH = SIGINP(ITH,II)
      LKVTOT = LKVL(KTH,1) + LKVL(KTH,2)
      NINT = NINT + LKVTOT
      LKDELT = LKDEL(KTH,1) + LKDEL(KTH,2)
      NDEL = NDEL + LKDELT
      DELVL1 = LKDEL(KTH,1) / FLOAT(LKVL(KTH,1))
```

```
      DELVL2 = LKDEL(KTH,2) / FLOAT(LKVL(KTH,2))
      DELVLT = LKDELT / FLOAT(LKVTOT)
 1250 WRITE(6,6) KTH,LKVL(KTH,1),LKVL(KTH,2),LKVTOT,LKMAXQ(KTH,1),
     1           LKMAXQ(KTH,2),LKDEL(KTH,1),LKDEL(KTH,2),LKDELT,
     2           DELVL1,DELVL2,DELVLT
      NTOT = NTOT + NINT
      NDTOT = NDTOT + NDEL
      DEL = NDEL / FLOAT(NINT)
      WRITE(6,7) NINT,DEL
 1275 CONTINUE
      DEL = NDTOT / FLOAT(NTOT)
      WRITE(6,8) DEL
      RETURN
      END
      SUBROUTINE STATA(I)
C
C     ..... PERIOD BY PERIOD OUTPUT GENERATOR.  LKSOUT IS AN ARRAY
C           TELLING THE DESIRED OUTPUT LINKS.
C
      INCLUDE DEFINS
      DIMENSION LKSOUT(4),N1(4),AVG1(4),X1(4),L1(4),LL1(4),LS1(4),LR1(4)
     1    ,DEL1(4),NL1(4),NS1(4),NR1(4),DELY1(4)
      DATA LKSOUT / 1,3,0,0 /
      PARAMETER N=2
      GO TO (10,20),I
   10 CONTINUE
      WRITE(6,4)
      DO 15 II=1,N
      L1(II)=0
      DEL1(II)=0
      LL1(II)=0
      LS1(II)=0
   15 LR1(II)=0
      RETURN
   20 WRITE(6,1) CLOCK,NUMVEH
      DO 30 II=1,N
      MM = LKSOUT(II)
      L1P = LKVL(MM,1) + LKVL(MM,2)
      N1(II)=L1P - L1(II)
      L1(II)=L1P
      NL1(II)=LKVD(MM,1)-LL1(II)
      LL1(II)=LKVD(MM,1)
      NS1(II)=LKVD(MM,2)-LS1(II)
      LS1(II)=LKVD(MM,2)
      NR1(II)=LKVD(MM,3)-LR1(II)
      LR1(II)=LKVD(MM,3)
      IDELP = LKDEL(MM,1) + LKDEL(MM,2)
      DELY1(II)=IDELP-DEL1(II)
      DEL1(II)=IDELP
      AVG1(II)=DELY1(II)/N1(II)
      X1(II)=IDELP/FLOAT(L1P)
   30 WRITE(6,2) L1P,(LKVD(MM,J),J=1,3),
     1  IDELP,X1(II),N1(II),NL1(II),NS1(II),NR1(II),AVG1(II)
      PUNCH 3,CLOCK,NUMVEH,((N1(II),AVG1(II),X1(II)),II=1,N)
      RETURN
    1 FORMAT(/,9H CLOCK = ,I7,13H    NUMVEH = ,I7)
    2 FORMAT(10X,I6,3I5,I8,F10.2,I10,3I5,F8.2)
```

```
    3 FORMAT(I7,I5,4(I5,2F6.1))
    4 FORMAT(1H1,12X,10HCUMULATIVE,18X,7HCURRENT,7X,11HLAST PERIOD,/,
     1  13X,25HVOL   LF   ST   RT  DELAY,7X,3HMOE,7X,
     2  25HVOL   LF   ST   RT  DEL/V)
      END
      FUNCTION TBAGEN(NTH)
C
C     ..... THIS FUNCTION GENERATES THE TIME BETWEEN ARRIVALS
C           AT THE NTH NODE.  AN EXPONENTIAL TIME BETWEEN
C           ARRIVALS IS ASSUMED.
C
      INCLUDE DEFINS
      TBAGEN = -NOPARS(NTH,1) * ALOG(RN1(NOSEED(NTH)))
      RETURN
      END
      SUBROUTINE VCHAIN(JTH,ITIME)
C
C     ..... THIS SUBROUTINE PUTS THE JTH VEHICLE ON THE
C           MOVE CHAIN.
C
      INCLUDE DEFINS
C
C     ..... IF CHAIN IS EMPTY, PUT AT HEAD.
C
      IF(MOVFST .NE. 0)  GO TO 10
C
      MOVFST = JTH
      MOVTIM = ITIME
      VEHNXT(JTH) = 0
      RETURN
C
C     ..... IF TIME FOR FIRST VEHICLE IS LESS THAN ITIME,
C           MUST BRACKET A PLACE ON THE CHAIN.
C
   10 IF(VTIME(MOVFST) .LT. ITIME)  GO TO 20
C
      VEHNXT(JTH) = MOVFST
      MOVFST = JTH
      MOVTIM = ITIME
      RETURN
C
C     ..... BRACKET A SPOT ON THE CHAIN.
C
   20 JTHLST = MOVFST
   25 JTHP = VEHNXT(JTHLST)
      IF(JTHP .EQ. 0)  GO TO 30
      IF(VTIME(JTHP) .GE. ITIME)  GO TO 30
      JTHLST = JTHP
      GO TO 25
C
C     ..... POSITION FOUND, PUT ON CHAIN.
C
   30 VEHNXT(JTHLST) = JTH
      VEHNXT(JTH) = JTHP
      IF(MOVTIM .GT. ITIME)  MOVTIM = ITIME
      RETURN
      END
```

```
      SUBROUTINE VEGEN
C
C     ..... THIS SUBROUTINE IS CALLED WHEN IT IS TIME TO CREATE
C           A VEHICLE TO INPUT INTO THE NETWORK.  VEGTIM IS THE
C           SCHEDULED ARRIVAL TIME OF THE NEXT UNBLOCKED ARRIVAL.
C           IF VEGFLG NOT EQUAL TO ZERO THEN A VEHICLE IS BLOCKED
C           FROM ENTERING.
C
      INCLUDE DEFINS
C**** WRITE(6,1)
    1 FORMAT(' VEGEN SUBROUTINE.')
      VEGFLG = 0
      NTHLST = 0
      NTH = VEGIND
C
C     ..... SET UP NEXT NODE TO SEARCH, THEN CHECK CAPACITY OF
C           THIS OUTPUT LINK.
C
  100 NTHNXT = NOGNXT(NTH)
      KTH = NOOUTP(NTH)
  110 IF(LKCAP(KTH) .LE. LKCONT(KTH))  GO TO 600
C
C     ..... FIND AN AVAILABLE VEHICLE AND FILL PARAMETERS.
C
      LKCONT(KTH) = LKCONT(KTH) + 1
      NUMVEH = NUMVEH + 1
      IF(NUMVEH .GT. MAXVEH) CALL DUMP(KTH,'VEGEN ','MAXVEH',27)
      DO 200 JTH=1,MAXVEH
      IF(VDSP(JTH) .EQ. 0)  GO TO 300
  200 CONTINUE
      CALL DUMP(0,'VEGEN ','MAXVEH',31)
  300 VDSP(JTH) = SPDDIS(ISEED)
      VSTATE(JTH) = 0
      VLK(JTH) = KTH
      VLKTIM(JTH) = CLOCK
      X = RN1(ISEED)
      DO 310 I=1,2
      IF(X .LE. LKPROB(KTH,I))  GO TO 320
  310 CONTINUE
      VTURN(JTH) = 3
      VLAN(JTH) = LKLANS(KTH)
      GO TO 350
  320 VTURN(JTH) = I
      IF(I .EQ. 2)  GO TO 330
      VLAN(JTH) = 1
      GO TO 350
  330 VLAN(JTH) = RN1(ISEED) * LKLANS(KTH) + 1
  350 ILANE = VLAN(JTH)
      VTIME(JTH) = LKLNG(KTH) * 10 / VDSP(JTH) + CLOCK
      IF(VTIME(JTH) .LT. LKARVT(KTH,ILANE) + 10)  VTIME(JTH) =
     1      LKARVT(KTH,ILANE) + 10
      VTIME(JTH) = VTIME(JTH) / 10 * 10
      LKARVT(KTH,ILANE) = VTIME(JTH)
      CALL VCHAIN(JTH,VTIME(JTH))
C
C     ..... SCHEDULE NEXT ARRIVAL AND PUT NODE BACK INTO CHAIN.
C
```

```
      NOCLK(NTH) = NOCLK(NTH) + TBAGEN(NTH)
C**** WRITE(6,2) NTH,JTH,KTH,VTIME(JTH),NOCLK(NTH)
    2 FORMAT('  FROM NODE ',I3,' MOVE VEH ',I4,' ONTO LINK ',I3,
     1  '.  VEH NEXT MOVE AT ',I8,' NEXT NEW VEH AT' F8.2)
C
C     ..... IF THIS WAS LAST ON THE CHAIN, NO CHANGE NECESSARY.
C
      IF(NTHNXT .EQ. 0)  GO TO 900
C
C     ..... CHECK FOR RELATIVE CHANGE IN ORDER.
C
      IF(NOCLK(NTH) .GT. NOCLK(NTHNXT))  GO TO 400
      IF(NOCLK(NTH) .LE. CLOCK)  GO TO 110
      GO TO 900
C
C     ..... MUST FIND NEW PLACE ON CHAIN
C
  400 IF(NTHLST .NE. 0)  GO TO 410
      VEGIND = NTHNXT
      GO TO 420
  410 NOGNXT(NTHLST) = NTHNXT
  420 NTHOLD = NTHNXT
  425 NTHNEW = NOGNXT(NTHOLD)
      IF(NTHNEW .EQ. 0)  GO TO 500
      IF(NOCLK(NTHNEW) .GE. NOCLK(NTH))  GO TO 500
      NTHOLD = NTHNEW
      GO TO 425
C
C     ..... PUT BETWEEN TWO EVENTS OR AT END OF CHAIN.
C
  500 NOGNXT(NTHOLD) = NTH
      NOGNXT(NTH) = NTHNEW
      GO TO 610
C
C     ..... CHECK NEXT NODE.
C
  600 NTHLST = NTH
  610 IF(NTHNXT .EQ. 0)  GO TO 900
      IF(NOCLK(NTHNXT) .GT. CLOCK)  GO TO 900
      NTH = NTHNXT
      GO TO 100
C
C     ..... SET FLAG AND FIND NEXT UNBLOCKED ARRIVAL.
C
  900 IF(NOCLK(VEGIND) .LE. CLOCK)  GO TO 925
C
C     ..... NOTHING DELAYED.
C
      VEGTIM = IFIX(NOCLK(VEGIND) + 9.999999 ) / 10 * 10
      RETURN
C
C     ..... SOME NODE WAS BLOCKED, FIND VEGTIM.
C
  925 VEGFLG = 1
      NTHNEW = NOGNXT(VEGIND)
  950 IF(NOCLK(NTHNEW) .GT. CLOCK)  GO TO 975
      NTHNEW = NOGNXT(NTHNEW)
```

```
      IF(NTHNEW .GT. 0)  GO TO 950
      VEGTIM = 2**30
      RETURN
  975 VEGTIM = IFIX(NOCLK(NTHNEW) + 9.9999999) / 10 * 10
      RETURN
      END
```

APPENDIX F

LINK VOLUMES FOR CLOSED NETWORK

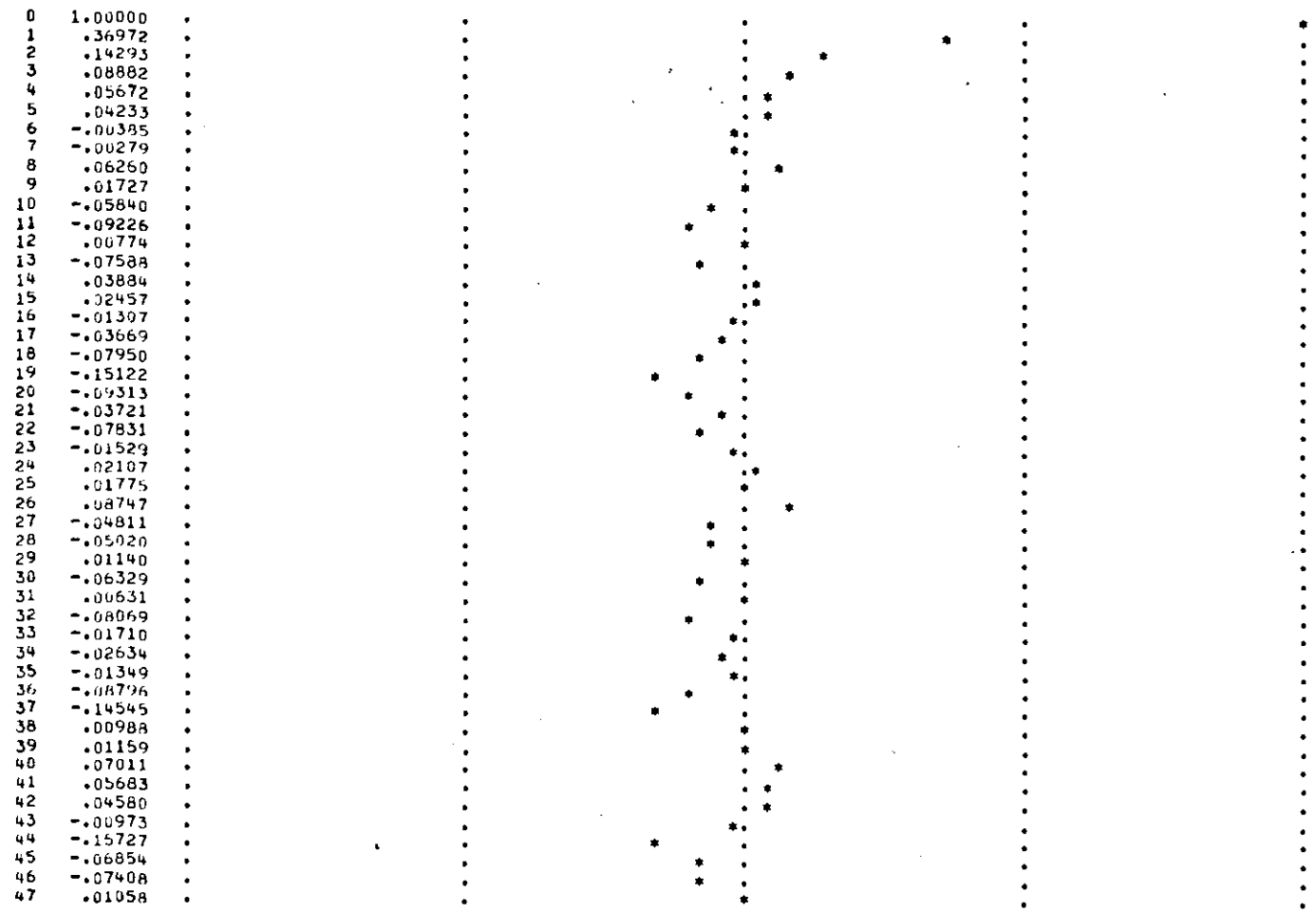
LIGHT FLOW				MEDIUM FLOW				HEAVY FLOW			
Link	Volume	Link	Volume	Link	Volume	Link	Volume	Link	Volume	Link	Volume
1	600.0	39	479.5	1	650.0	39	554.9	1	750.0	39	677.0
2	538.5	40	594.0	2	663.6	40	697.6	2	803.9	40	803.3
3	550.0	41	550.3	3	700.0	41	707.9	3	900.0	41	917.9
4	460.5	42	478.3	4	695.9	42	722.0	4	876.0	42	900.0
5	450.0	43	518.9	5	600.0	43	673.4	5	750.0	43	879.0
6	480.6	44	505.7	6	658.4	44	762.6	6	823.1	44	944.1
7	250.0	45	486.4	7	400.0	45	636.1	7	600.0	45	835.3
8	254.9	46	500.3	8	385.1	46	753.2	8	551.3	46	921.0
9	250.0	47	483.3	9	400.0	47	637.1	9	600.0	47	841.6
10	322.8	48	499.4	10	476.9	48	751.1	10	669.6	48	912.3
11	250.0	49	442.6	11	400.0	49	599.3	11	600.0	49	763.1
12	304.7	50	466.3	12	457.0	50	630.5	12	649.6	50	773.7
13	250.0	51	437.7	13	400.0	51	598.5	13	600.0	51	770.6
14	296.4	52	479.8	14	441.8	52	642.1	14	626.1	52	775.7
15	250.0	53	421.2	15	400.0	53	579.9	15	600.0	53	752.8
16	267.1	54	480.8	16	410.5	54	637.0	16	589.4	54	757.6
17	500.0	55	421.4	17	650.0	55	584.0	17	750.0	55	763.7
18	422.6	56	508.0	18	591.7	56	668.4	18	782.1	56	785.9
19	500.0	57	241.1	19	750.0	57	375.2	19	900.0	57	548.4
20	466.5	58	253.1	20	620.6	58	387.6	20	825.4	58	562.3
21	600.0	59	249.4	21	700.0	59	389.6	21	800.0	59	571.5
22	499.2	60	303.8	22	588.0	60	454.1	22	727.5	60	646.7
23	250.0	61	270.6	23	400.0	61	423.0	23	600.0	61	620.9
24	227.1	62	307.9	24	347.9	62	466.6	24	506.3	62	670.9
25	250.0	63	262.9	25	400.0	63	412.1	25	600.0	63	606.0
26	293.6	64	282.3	26	437.3	64	426.7	26	621.8	64	614.8
27	250.0	65	237.6	27	400.0	65	374.5	27	600.0	65	553.9
28	293.4	66	264.0	28	434.1	66	411.2	28	618.4	66	597.6
29	250.0	67	268.4	29	400.0	67	408.8	29	600.0	67	583.6
30	302.5	68	262.8	30	443.8	68	403.0	30	622.4	68	591.3
31	250.0	69	277.1	31	400.0	69	424.3	31	600.0	69	609.4
32	287.7	70	272.8	32	420.1	70	413.6	32	586.3	70	601.8
33	563.0	71	272.2	33	621.2	71	418.6	33	728.5	71	609.0
34	539.2	72	282.4	34	657.5	72	434.4	34	788.2	72	638.3
35	530.3	73	273.8	35	596.1	73	422.6	35	710.0	73	614.4
36	554.7	74	266.7	36	665.6	74	408.5	36	784.5	74	596.5
37	480.0	75	240.0	37	545.3	75	372.6	37	655.1	75	546.1
38	558.0	76	249.5	38	664.7	76	394.0	38	777.5	76	586.0

APPENDIX G

AUTOCORRELATION FUNCTION OUTPUT

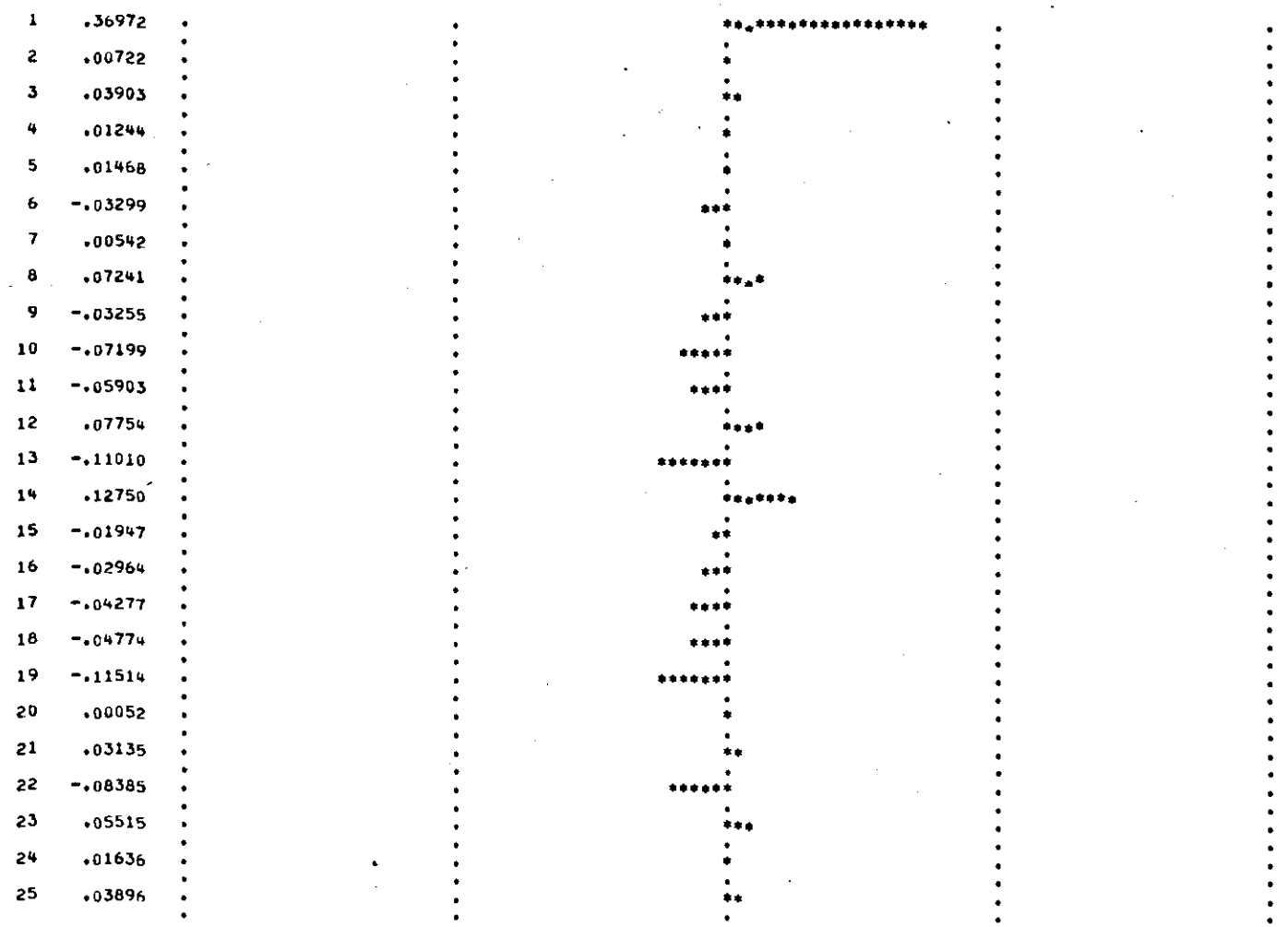
SAMPLE AUTOCORRELATION FUNCTION: CONT MODEL - LINK 1

FORMAT SPECIFICATION USED WAS (17X,F6.0)



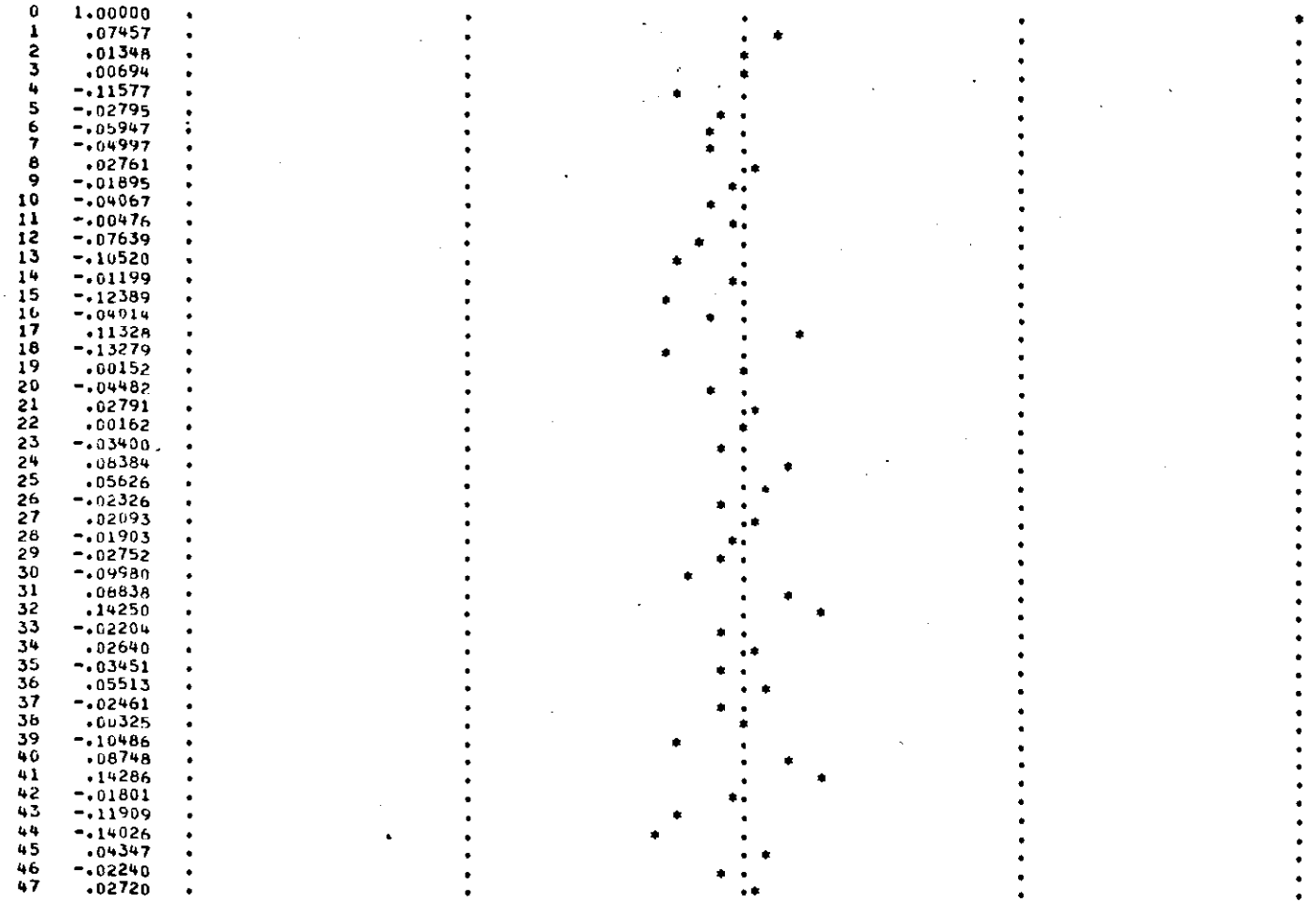
0	1.00000
1	.36972
2	.14293
3	.08882
4	.05672
5	.04233
6	-.00385
7	-.00279
8	.06260
9	.01727
10	-.05840
11	-.09226
12	.00774
13	-.07588
14	.03884
15	.02457
16	-.01307
17	-.03669
18	-.07950
19	-.15122
20	-.09313
21	-.03721
22	-.07831
23	-.01529
24	.02107
25	.01775
26	.08747
27	-.04811
28	-.05020
29	.01140
30	-.06329
31	.00631
32	-.08069
33	-.01710
34	-.02634
35	-.01349
36	-.08796
37	-.14545
38	.00988
39	.01159
40	.07011
41	.05683
42	.04580
43	-.00973
44	-.15727
45	-.06854
46	-.07408
47	.01058

PARTIAL AUTOCORRELATION FUNCTION: CONT MODEL - LINK 1



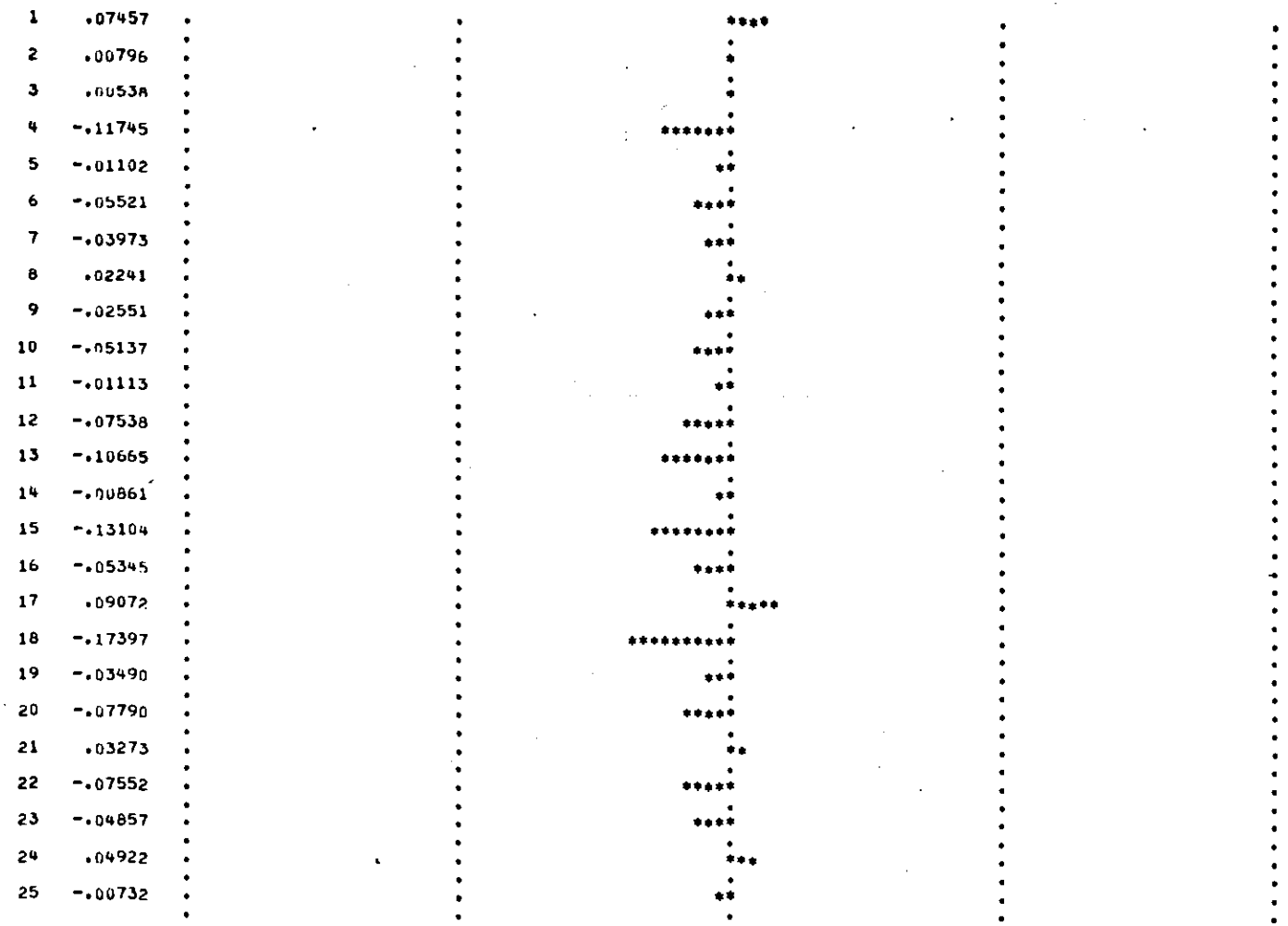
SAMPLE AUTOCORRELATION FUNCTION: CONT MODEL - LINK 5

FORMAT SPECIFICATION USED WAS (34X,F6.0)



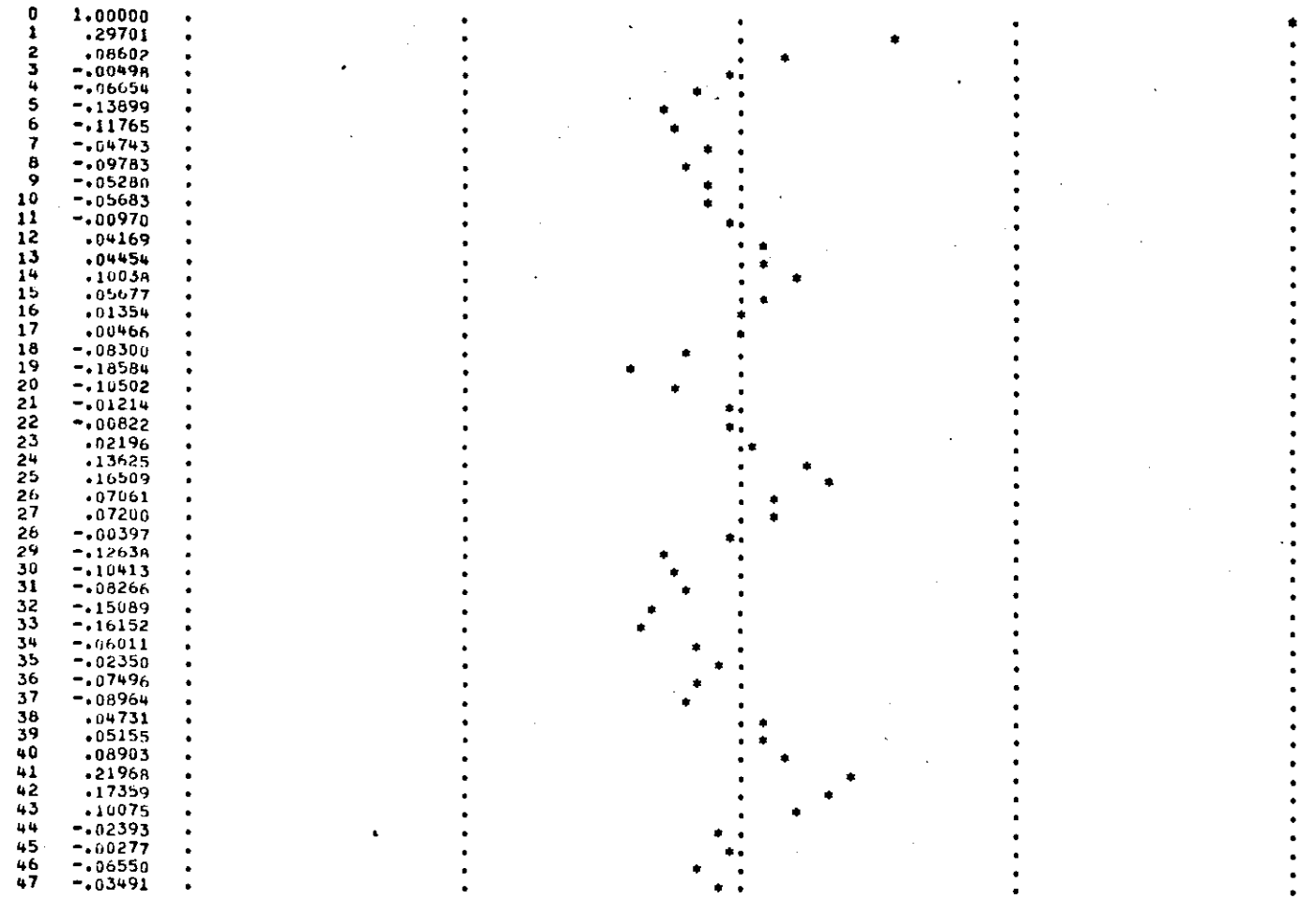
0	1.00000
1	.07457
2	.01348
3	.00694
4	-.11577
5	-.02795
6	-.05947
7	-.04997
8	.02761
9	-.01895
10	-.04067
11	-.00476
12	-.07639
13	-.10520
14	-.01199
15	-.12389
16	-.04014
17	.11328
18	-.13279
19	.00152
20	-.04482
21	.02791
22	.00162
23	-.03400
24	.08384
25	.05626
26	-.02326
27	.02093
28	-.01903
29	-.02752
30	-.09980
31	.08838
32	.14250
33	-.02204
34	.02640
35	-.03451
36	.05513
37	-.02461
38	.00325
39	-.10486
40	.08748
41	.14286
42	-.01801
43	-.11909
44	-.14026
45	.04347
46	-.02240
47	.02720

PARTIAL AUTOCORRELATION FUNCTION: CONT MODEL - LINK 5



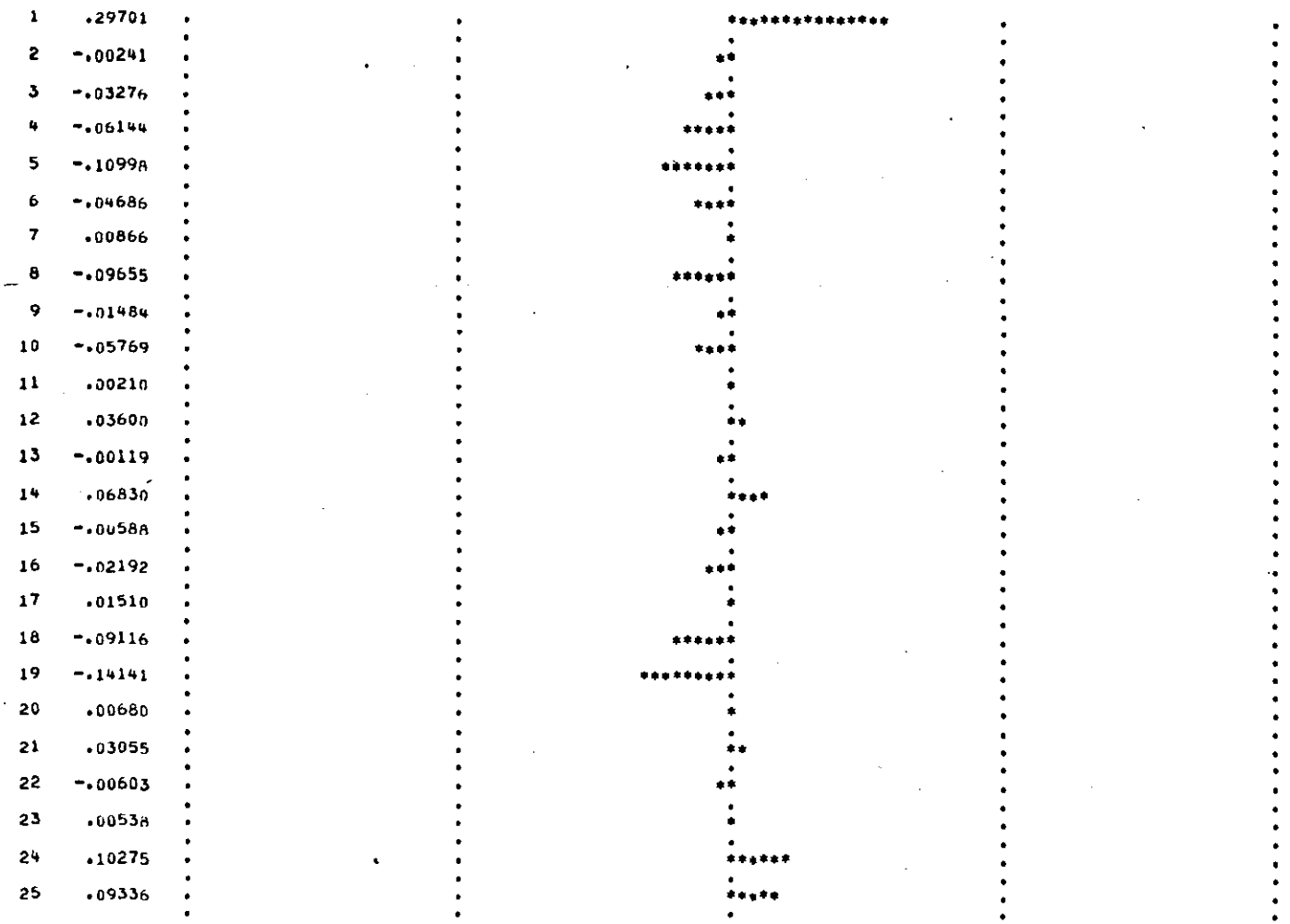
SAMPLE AUTOCORRELATION FUNCTION: UR MODEL - LINK 1

FORMAT SPECIFICATION USED WAS (17X,F6.0)



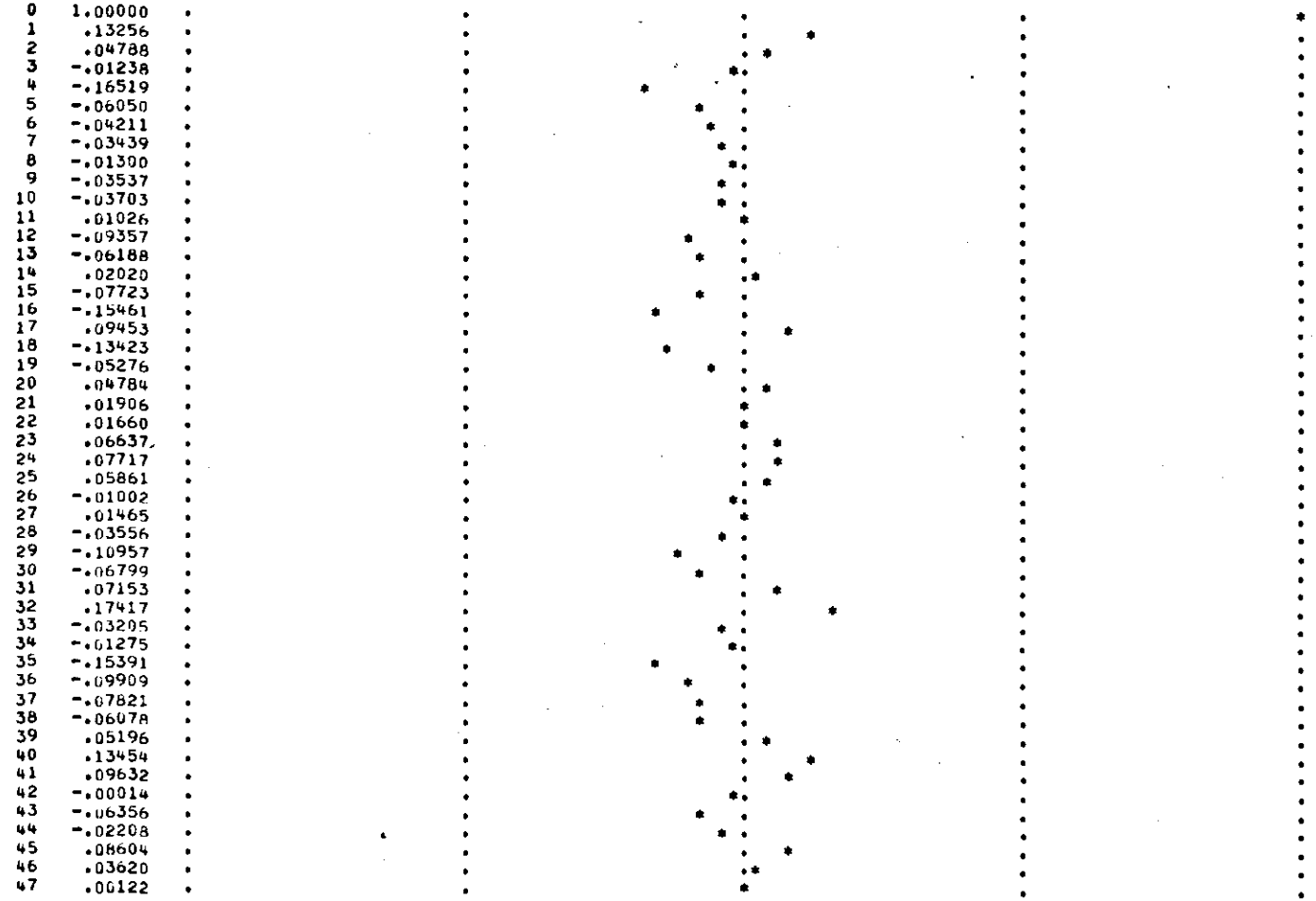
0	1.00000
1	.29701
2	.08602
3	-.00498
4	-.06654
5	-.13899
6	-.11765
7	-.04743
8	-.09783
9	-.05280
10	-.05683
11	-.00970
12	.04169
13	.04454
14	.10038
15	.05677
16	.01354
17	.00466
18	-.08300
19	-.18584
20	-.10502
21	-.01214
22	-.00822
23	.02196
24	.13625
25	.16509
26	.07061
27	.07200
28	-.00397
29	-.12638
30	-.10413
31	-.08266
32	-.15089
33	-.16152
34	-.06011
35	-.02350
36	-.07496
37	-.08964
38	.04731
39	.05155
40	.08903
41	.21968
42	.17359
43	.10075
44	-.02393
45	-.00277
46	-.06550
47	-.03491

PARTIAL AUTOCORRELATION FUNCTION: UB MODEL - LINK 1



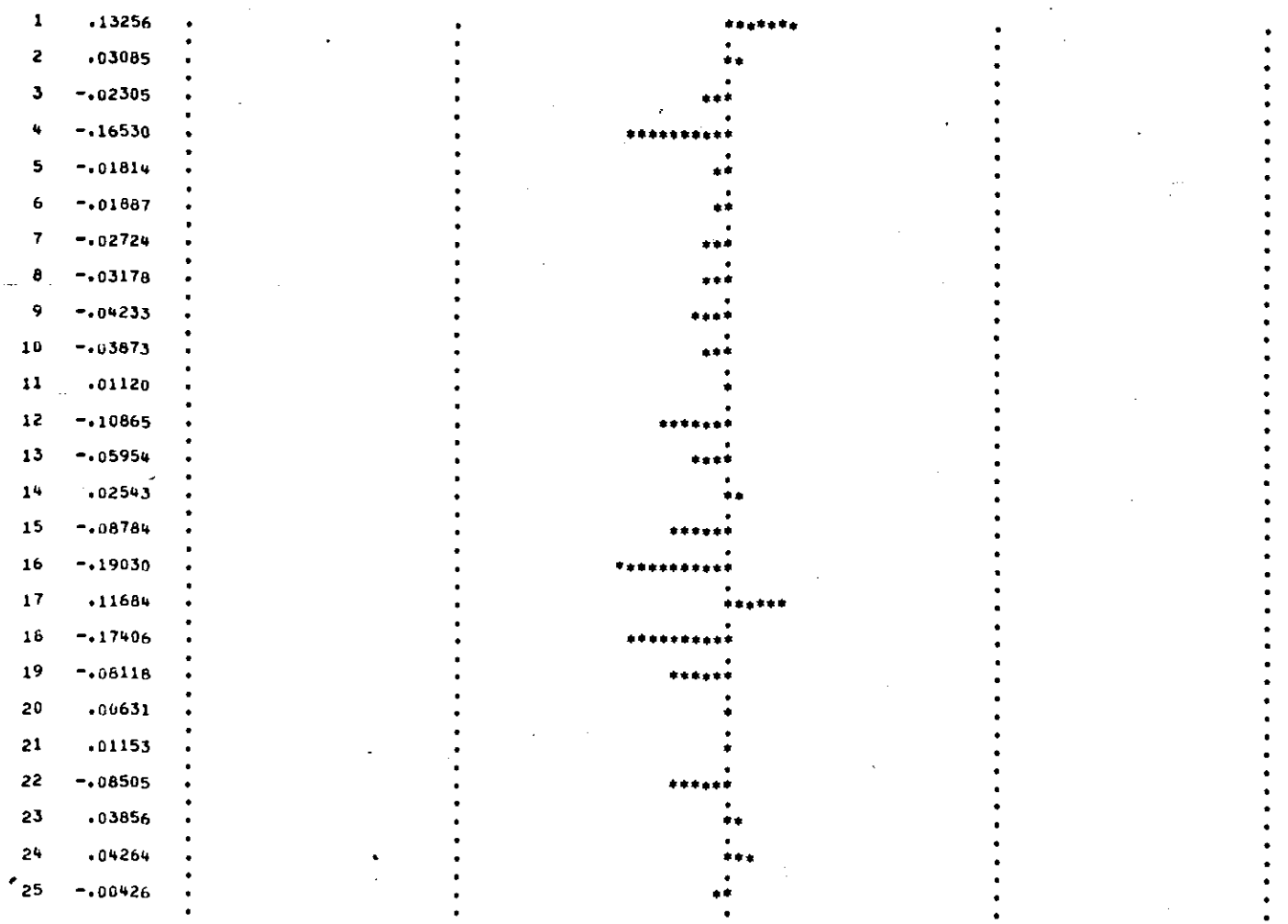
SAMPLE AUTOCORRELATION FUNCTION: UB MODEL - LINK 5

FORMAT SPECIFICATION USED WAS (34X,F6.0)



0	1.00000
1	.13256
2	.04788
3	-.01238
4	-.16519
5	-.06050
6	-.04211
7	-.03439
8	-.01300
9	-.03537
10	-.03703
11	.01026
12	-.09357
13	-.06188
14	.02020
15	-.07723
16	-.15461
17	.09453
18	-.13423
19	-.05276
20	.04784
21	.01906
22	.01660
23	.06637
24	.07717
25	.05861
26	-.01002
27	.01465
28	-.03556
29	-.10957
30	-.06799
31	.07153
32	.17417
33	-.03205
34	-.01275
35	-.15391
36	-.09909
37	-.07821
38	-.06078
39	.05196
40	.13454
41	.09632
42	-.00014
43	-.06356
44	-.02208
45	.08604
46	.03620
47	.00122

PARTIAL AUTOCORRELATION FUNCTION: U8 MODEL - LINK 5

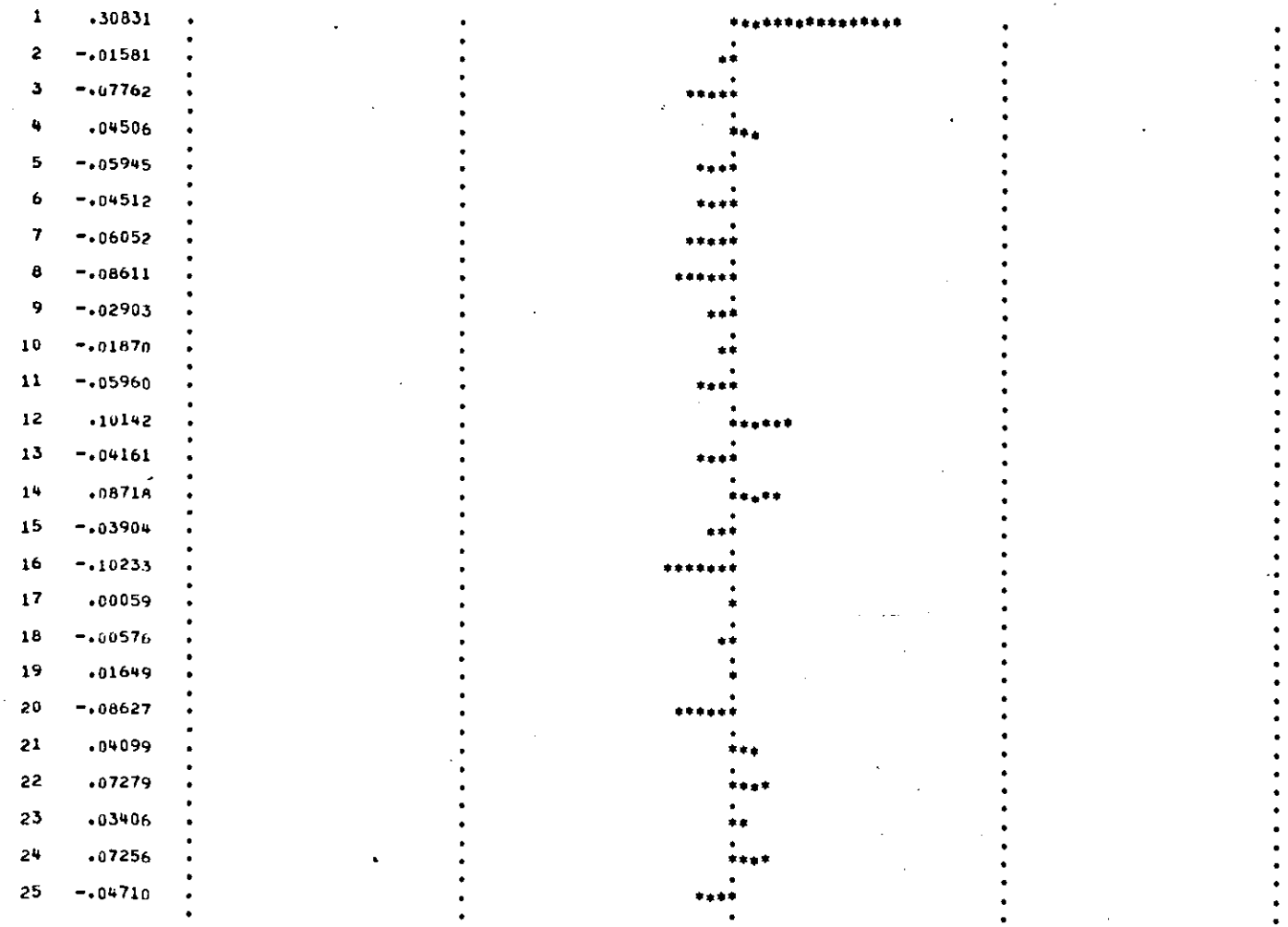


SAMPLE AUTOCORRELATION FUNCTION: ZONE MODEL - LINK 1

FORMAT SPECIFICATION USED WAS (17X,F6.0)

0	1.00000	*
1	.30831	*
2	.08074	*
3	-.04982	*
4	.00173	*
5	-.04672	*
6	-.06384	*
7	-.09531	*
8	-.12071	*
9	-.08773	*
10	-.04917	*
11	-.06079	*
12	.06369	*
13	.02114	*
14	.10255	*
15	.02082	*
16	-.05875	*
17	-.05344	*
18	-.01242	*
19	.01979	*
20	-.07706	*
21	-.01708	*
22	.05173	*
23	.07570	*
24	.11443	*
25	.01384	*
26	.08320	*
27	.12429	*
28	.14316	*
29	.05181	*
30	-.11850	*
31	-.06271	*
32	-.05231	*
33	-.03357	*
34	-.06946	*
35	.07219	*
36	.00446	*
37	-.04594	*
38	-.01401	*
39	.09210	*
40	.11320	*
41	.03027	*
42	-.05257	*
43	-.06401	*
44	-.04143	*
45	.01608	*
46	-.07497	*
47	-.15465	*

PARTIAL AUTOCORRELATION FUNCTION: ZONE MODEL - LINK 1

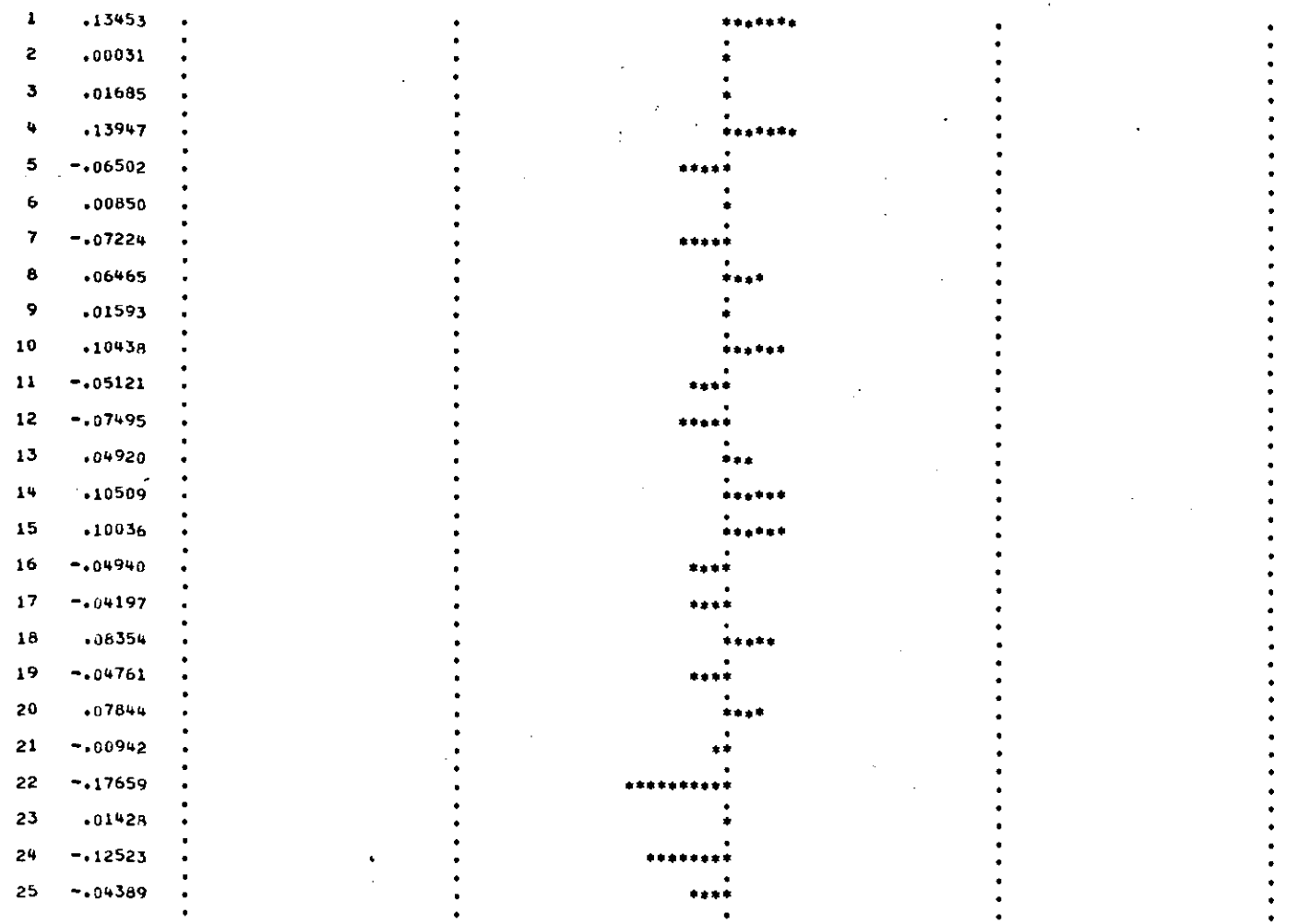


SAMPLE AUTOCORRELATION FUNCTION: ZONF MODEL - LINK 5

FORMAT SPECIFICATION USED WAS (34X,F6.0)

0	1.00000	.
1	.13453	.
2	.01840	.
3	.01906	.
4	.14170	.
5	-.02517	.
6	-.00173	.
7	-.06693	.
8	.06222	.
9	.01944	.
10	.10652	.
11	-.03732	.
12	-.05823	.
13	.02835	.
14	.14025	.
15	.09421	.
16	-.02759	.
17	-.03803	.
18	.11922	.
19	-.00106	.
20	.03906	.
21	-.01762	.
22	-.13784	.
23	-.01282	.
24	-.07182	.
25	-.09977	.
26	-.03490	.
27	-.00086	.
28	-.00493	.
29	.01032	.
30	-.10451	.
31	-.22931	.
32	-.05260	.
33	-.04016	.
34	-.08890	.
35	-.06000	.
36	-.05849	.
37	-.07238	.
38	-.00445	.
39	-.16230	.
40	-.04761	.
41	-.13660	.
42	-.08183	.
43	-.07060	.
44	-.08076	.
45	-.15723	.
46	-.04180	.
47	.11655	.

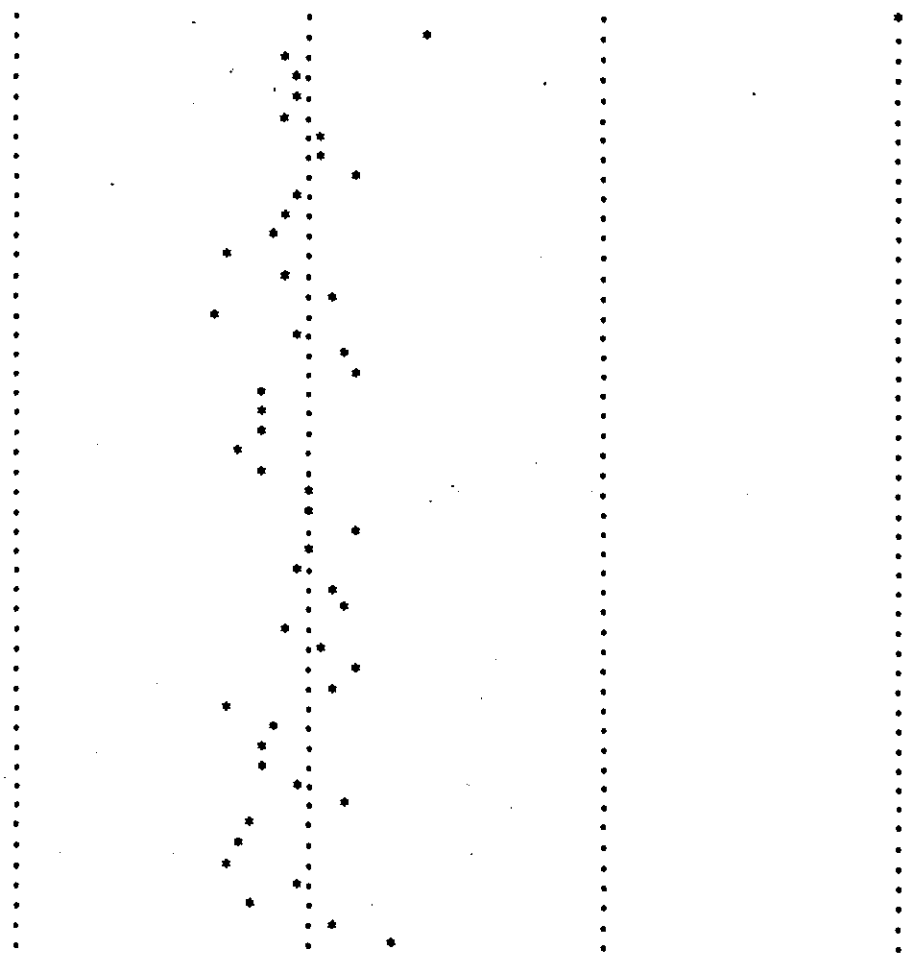
PARTIAL AUTOCORRELATION FUNCTION: ZONE MODEL - LINK 5



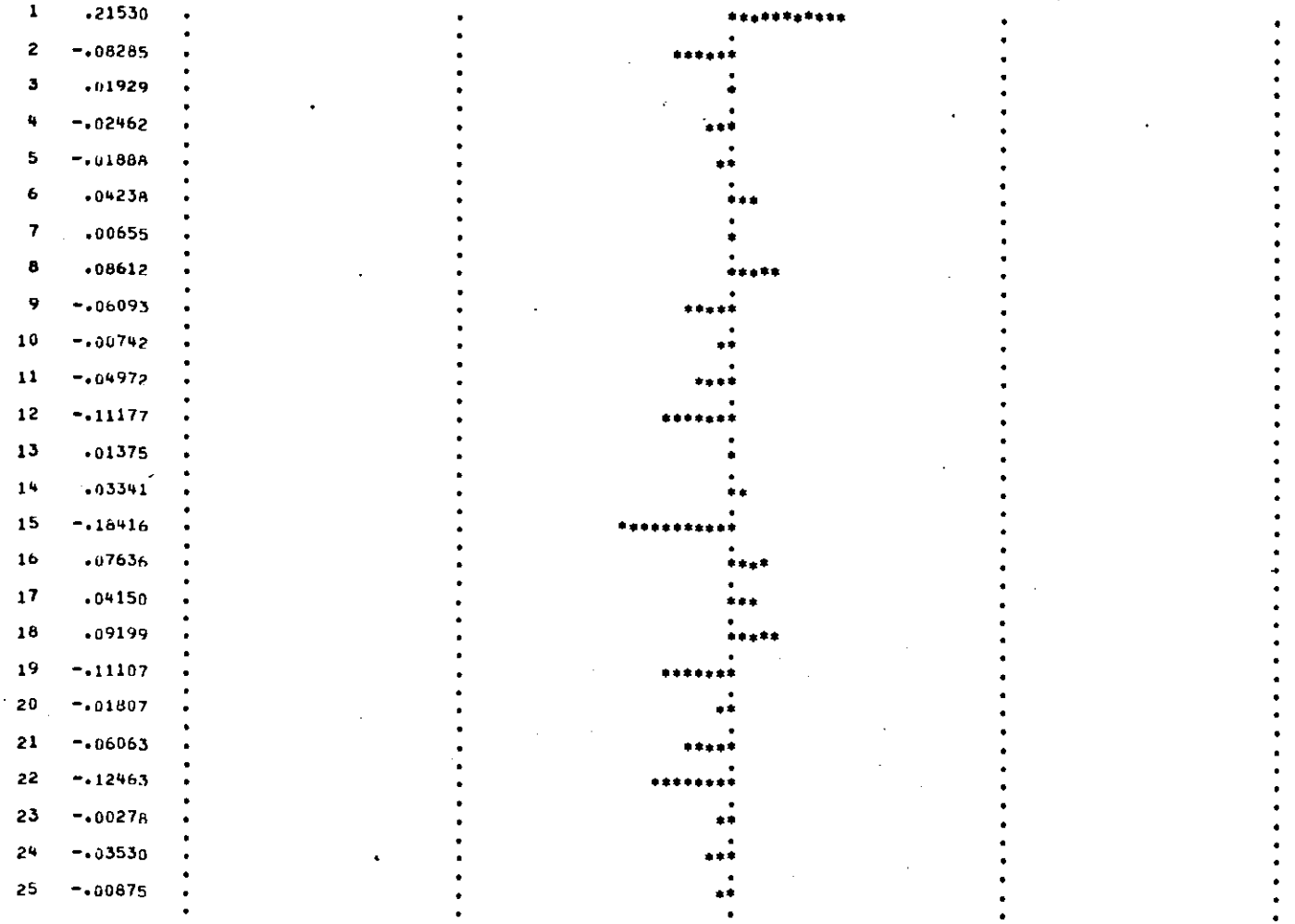
SAMPLE AUTOCORRELATION FUNCTION: NEXT MODEL = LINK 1

FORMAT SPECIFICATION USED WAS (17X,F6.0)

0	1.00000	.
1	.21530	.
2	-.03266	.
3	-.00718	.
4	-.01799	.
5	-.02757	.
6	.03164	.
7	.02585	.
8	.08456	.
9	-.01923	.
10	-.03546	.
11	-.05363	.
12	-.12656	.
13	-.03955	.
14	.04846	.
15	-.15007	.
16	-.00460	.
17	.07035	.
18	.09411	.
19	-.07010	.
20	-.07236	.
21	-.07293	.
22	-.11997	.
23	-.07366	.
24	.00539	.
25	.00680	.
26	.08995	.
27	.00731	.
28	-.00741	.
29	.05710	.
30	.06230	.
31	-.02729	.
32	.02422	.
33	.08474	.
34	.05261	.
35	-.13631	.
36	-.05469	.
37	-.06755	.
38	-.06086	.
39	-.01018	.
40	.06071	.
41	-.09793	.
42	-.10287	.
43	-.12202	.
44	-.00024	.
45	-.08189	.
46	.05092	.
47	.14182	.

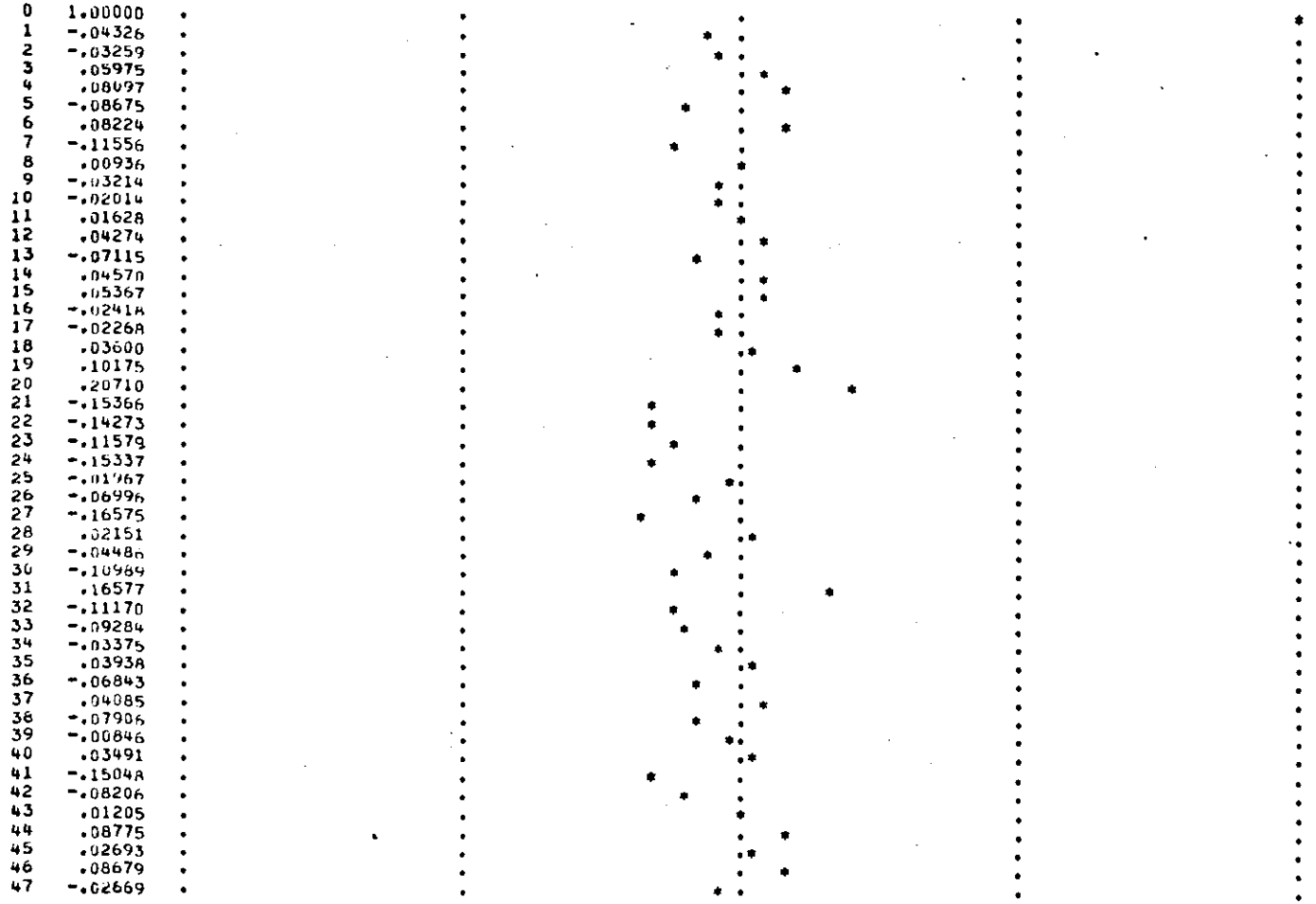


PARTIAL AUTOCORRELATION FUNCTION: NEXT MODEL - LINK 1

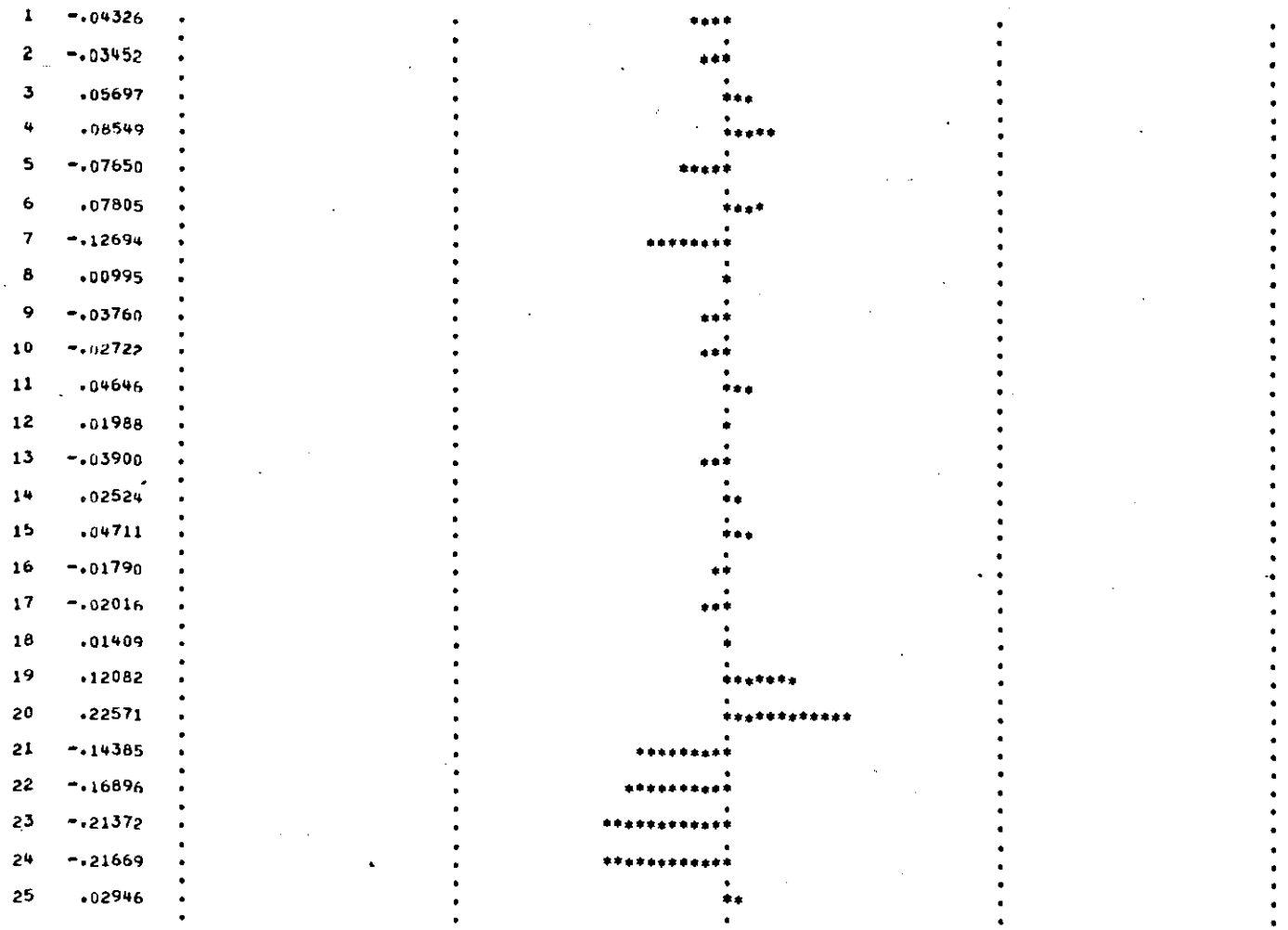


SAMPLE AUTOCORRELATION FUNCTION: NEXT MODEL - LINK 5

FORMAT SPECIFICATION USED WAS (34X,F6.0)



PARTIAL AUTOCORRELATION FUNCTION: NEXT MODEL - LINK 5



BIBLIOGRAPHY OF CITED LITERATURE

1. Barnes, J. W., "A Simulated Study of Traffic Intersection Control," Ph.D. Dissertation, University of Arkansas, 1971.

2. Benhard, F. G., "Simulation of a Traffic Intersection on a Digital Computer," M.S. Thesis, University of California, Los Angeles, 1959.

3. Berry, D. S., "Field Measurement of Delay at Signalized Intersections," *Proceedings, Highway Research Board, 35,* 1956, pp. 505-522.

4. Blum, A. M., "A General Purpose Digital Simulator and Examples of Its Application," *IBM Systems Journal,* Vol. 3, No. 1, 1964, pp. 41-47.

5. Blum, A. M., "A General Purpose Digital Traffic Simulator," *Simulation,* Vol. 14, No. 1, January, 1970, pp. 9-25.

6. Box, G. E. P. and Jenkins, G. M., *Time Series Analysis, Forecasting, and Control,* Holden-Day, Inc., San Francisco, 1970.

7. Buhr, J. H., Messerole, T. C. and Drew, D. R., "A Digital Simulation Program of a Section of Freeway with Entrance and Exit Ramps," *Highway Research Record 230,* Highway Research Board, 1968, pp. 15-31.

8. Cassel, A. and Janoff, M. S., "A Simulation Model of a Two-Lane Rural Road," *Highway Research Record 257,* Highway Research Board, 1968, pp. 1-16.

9. Chandler, R. E., Herman, R. and Montroll, E., "Traffic Dynamics: Studies in Car Following," *Operations Research,* Vol. 6, No. 2, 1958, pp. 165-184.

10. Collins, J. T., "Development of Volume-Speed Delay Model by Simulation of Roadway Network Analysis," Ph.D. Dissertation, New York University, 1971.

11. Francis, J. G. F. and Lott, R. S., "A Simulation Programme for Linked Traffic Signals," paper presented at the Second International Symposium on the Theory of Traffic Flow, London, June 25-27, 1963.

12. Gerlough, D. L., "Simulation of Freeway Traffic on a General-Purpose Discrete Variable Computer," Ph.D. Dissertation, University of California, Los Angeles, 1955.

13. Gerlough, D. L., and Wagner, F. A., "Simulation of the Network," *Proceedings, Conference on Traffic Surveillance, Simulation and Control,* Washington, D. C., Sept. 14-15, 1964, pp. 152-165.

14. Gerlough, D. L., "Simulation of Traffic Flow" in "An Introduction to Traffic Flow Theory," *Highway Research Board Special Report 79,* 1964.

15. Gerlough, D. L. and Wagner, F. A., "Improved Criteria for Designing and Timing Traffic Signal Systems," Final Report on NCHRP Project 3-5, Planning Research Corp., March, 1964.

16. Gerlough, D. L., and Wagner, F. A., "Improved Criteria for Traffic Signals at Individual Intersections," *National Cooperative Highway Research Program Report 32,* Highway Research Board, 1967.

17. Glickstein, A., Findley, L. D. and Levy, S. L., "A Study of the Application of Computer Simulation Techniques to Interchange Design Problems," Final Report for the U. S. Bureau of Public Roads, Midwest Research Institute, 1960. Also in *Highway Research Board, Bulletin 291,* 1961, pp. 139-162.

18. Goode, H. H., Pollmar, C. H. and Wright, J. B., "The Use of a Digital Computer to Model a Signalized Intersection," *Proceedings Highway Research Board 35,* 1956, pp. 548-557.

19. Goode, H. H. and True, W. C., "Simulation and Display of Four Inter-Related Vehicular Traffic Intersections," *13th National Meeting of the Association for Computing Machinery Proceedings,* 1958.

20. Highway Research Board, *Highway Capacity Manual, 1965,* Special Report 87, Highway Research Board, 1966.

21. Howat, M. G., "A Digital Computer Simulation of Driver Overtaking, Following, and Passing," CAL Report No. VK-1938-V-I, Cornell Aeronautical Lab., Inc., Buffalo, N. Y., July 15, 1965.

22. Katz, J. H., "Simulation of a Traffic Network," *Communications of the ACM,* Vol. 6, No. 8, 1963, pp. 480-486.

23. Kell, J. H., "Intersection Delay Obtained by Simulating Traffic on a Computer," *Highway Research Record 15,* Highway Research Board, 1963, pp. 73-97.

24. Kell, J. H., "Simulation of the Intersection," *Proceedings, Conference on Traffic Surveillance, Simulation, and Control*, Washington, D. C., September 14-15, 1964, pp. 127-143.

25. Kell, J. H., "Traffic Simulation Validation," *Proceedings, Program Review Meeting, Research and Development of Traffic Systems*, Gaithersburg, Maryland, December, 1966, pp. 371-379.

26. Kell, J. H., "Intersection Simulation Model Validation," *Final Report*, Traffic Research Corporation, San Francisco, 1966.

27. Lewis, R. M., "The Simulation of Vehicular Traffic at an Intersection on a Digital Electronic Computer," M.S. Thesis, Rensselear Polytechnic Institute, 1959.

28. Lewis, R. M., "The Simulation of Traffic Flow to Obtain Volume Warrants for Intersection Control," Ph.D. Dissertation, Purdue University, August, 1962. Reprinted as report for Joint Highway Research Project, No. 23, Purdue University, Lafayette, Indiana, 1962.

29. Lieberman, E. B., Worrall, R. D. and Bruggeman, J. M., "Logic Design and Demonstration of UTCS-1 Network Simulation Model," *Highway Research Record 409*, Highway Research Board, 1972, pp. 46-56.

30. Morgan, J. T. and Little, J. D. C., "Synchronizing Traffic Signals for Maximal Bandwidth," *Operations Research*, Vol. 12, No. 6, 1964, pp. 896-912.

31. Perchonok, P. A. and Levy, S. L., "Application of Digital Computer Simulation Techniques to Freeway On-Ramp Traffic Operations," *Proceedings, Highway Research Board 39*, Highway Research Board, 1960, pp. 506-523.

32. Rhee, S. Y., "An Urban Traffic Control Simulator," paper presented at the Second International Symposium on the Theory of Traffic Flow, London, June 25-27, 1963.

33. Sakai, T. and Nagao, M. "Simulation of Traffic Flow in a Network," *Communications of the ACM*, Vol. 12, No. 6, June 1969, pp. 311-318.

34. Schwartz, J. G., "An Urban Street Network Computer Simulation Model," Research Report R66-52, Civil Engineering Systems Laboratory , MIT, Cambridge, Mass., September, 1966.

35. Stark, M. C., "Computer Simulation of Nine Blocks of a City Street," *Highway Research Board*, Bulletin 356, 1962, pp. 40-47.

36. Thomasson, J. N., Jr., "Simulation of Traffic at a Two-Way Stop Intersection," M.S. Thesis, Georgia Institute of Technology, Atlanta, 1966.

37. Voskoglov, N. and Wheeler, P. J., "Optimizing a Local Street System by Simulation," *Traffic Quarterly*, Vol. 23, No. 2, April, 1969, pp. 179-196.

38. Wagner, F. A., Gerlough, D. L. and Barnes, F. C., "Improved Criteria for Designing and Timing Traffic Signal Systems: Urban Arterials," *National Cooperative Highway Research Program Report 73*, Highway Research Board, 1969.

39. Wagner, F. A., Barnes, F. C. and Gerlough, D. L., "Improved Criteria for Traffic Signal Systems in Urban Networks," *National Cooperative Highway Research Program Report 124*, Highway Research Board, 1971.

40. Webster, F. V., "Traffic Signal Settings," Road Research Laboratory Technical Paper No. 39, HMSO, London, England, 1958.

41. Wohl, M., "Simulation--Its Application to Traffic Engineering, Part II," *Traffic Engineering*, Vol. 31, No. 1, October, 1960, pp. 19-25, 56.

42. Wong, S. R., "Traffic Simulator with a Digital Computer," *Proceedings, Western Joint Computer Conference*, San Francisco, California, February, 7-9, 1956, pp. 92-94.

43. Wright, P. H., "Simulation of Traffic at a Four-way Stop Intersection," Final Report, Project B-604, School of Civil Engineering, Georgia Institute of Technology, October, 1967.

BIBLIOGRAPHY ON SIMULATION OF TRAFFIC FLOW

Aitken, J. M., "Simulation of Traffic Conditions at an Uncontrolled T-Junction," *Traffic Engineering and Control*, Vol. 5, No. 6, October, 1963, pp. 354-358.

Baker, R. L., "Developments in Traffic Simulation and Control," *Public Works*, Vol. 95, No. 3, March, 1964, pp. 98-101.

Baker, R. L., "The Effects of Feedback on Freeway Operations," M.S. Thesis, Texas A & M University, 1967.

Banks, J. N., "Traffic on a Two-Lane, Two-Way Rural Road," M.S. Thesis, Georgia Institute of Technology, Atlanta, 1969.

Barnes, F. C., "An Investigation to Determine Possible Warrants for the Construction of Grade Separations in Lieu of Traffic Signals of Major Urban Street Intersections," M.S. Thesis, University of California, Los Angeles, 1968.

Barnes, F. C. and Wagner, F. A., Jr., "Case Study in the Application of a Traffic Network Simulation Model," paper presented at the 34th National ORSA Meeting, Philadelphia, 1968.

Barnes, J. W., "A Simulated Study of Traffic Intersection Control," Ph.D. Dissertation, University of Arkansas, 1971.

Barnes, J. W. and Crisp, R. M., Jr., "The Study of the Traffic Intersection Through Simulation," presented at the 40th National ORSA Meeting, Anaheim, California, 1971.

Beilby, M. H., "Traffic Simulation by Digital Computer," *Traffic Engineering and Control*, Vol. 10, No. 1, May, 1968, pp. 21-23, 27.

Bellis, W. R., "Traffic Flow Simulation Through Successive Traffic Signals," paper presented at the Joint ORSA-TIMS Meeting, San Francisco, 1968.

Bendtsen, P. H., "Research at the Technical University of Denmark," *Traffic Quarterly*, Vol. 23, No. 3, July, 1969, pp. 413-426.

Benhard, F. G., "Simulation of a Traffic Intersection on a Digital Computer," M.S. Thesis, University of California, Los Angeles, 1959.

Benson, J. C., "A Programme to Simulate on a Pegasus II Computer the Behavior of Traffic at a Single Intersection Controlled by Vehicle-Actuated Traffic Signals," Unpublished Laboratory Note LN/182/JCB, Department of Science and Industrial Research, Road Research Laboratory, 1960.

Bleyl, R. L., "Simulation of Traffic Flow to Compare Regular and Flashing Signals," *Proceedings, Institute of Traffic Engineers,* 1964, pp. 153-161.

Blum, A. M., "A General Purpose Digital Simulator and Examples of Its Application," *IBM Systems Journal,* Vol. 3, No. 1, 1964, pp. 41-47.

Blum, A. M., "Vehicle Traffic Simulator Program," SHARE General Program Library 7090 IBM, 0027 IBM Technical Information Exchange, August, 1965.

Blum, A. M., "A General Purpose Digital Traffic Simulator," *Simulation,* Vol. 14, No. 1, January, 1970, pp. 9-25.

Braunstein, M. L., Laughery, K. R., and Siegfried, J. B., "Computer Simulation of the Automobile Driver: A Model of the Car Follower," *Highway Research Record 55,* Highway Research Board, 1963, pp. 21-28.

Braunstein, M. L., Laughery, K. R. and Siegfried, J. B., "Computer Simulation of Driver Behavior During Car Following: A Methodology Study," CAL Report No. YM-1797-H-1, Cornell Laboratory, Inc., Buffalo, New York, 1963.

Bruggeman, J. M., Lieberman, E. B. and Worrall, R. D., "Network Flow Simulation for Urban Traffic Control Systems," KLD Assoc., Inc., Technical Report, FH-11-7462-2, 1971.

Buhr, J. H., Messerole, T. C. and Drew, D. R., "A Digital Simulation Program of a Section of Freeway with Entrance and Exit Ramps," *Highway Research Record 230,* Highway Research Board, 1968, pp. 15-31.

Bullen, A. G. R., "Field Test of an Intersection Simulation Model," M.S. Thesis, Northwestern University, 1964.

Cassel, A., "Remedial Aids of Overtaking and Passing on Two-Lane Rural Highways," *Proceedings, Program Review Meeting, Research and Development of Traffic Systems,* Gaithersburg, Md., December, 1966, pp. 27-42.

Cassel, A. and Janoff, M. S., "A Simulation Model of a Two-Lane Rural Road," *Highway Research Record 257,* Highway Research Board, 1968, pp. 1-16.

Collins, J. T., "Development of Volume-Speed Delay Model by Simulation of Roadway Network Analysis," Ph.D. Dissertation, New York University, 1971.

Cooper, D. L., Knox, R. M. and Walinchos, R. J., "Final Report: Systems Analysis Methodology in Urban Traffic Control Systems," TRW Systems Group Report 11644-H 014-RO-00, Houston, June, 1969.

Cosgriff, R. L., "Transportation Simulation," *Proceedings, Conference on Traffic Surveillance, Simulation, and Control,* Washington, D. C., Sept. 14-15, 1964, pp. 123-126.

Culshaw, T. A., Morris, R. W. J. and Pak-Poy, P. G., "Comparison of Traffic Signal Controllers Using Computer Simulation--Some Preliminary Results," *Australian Road Research Board,* Vol. 3, Part 1, 1966, pp. 471-487.

Dart, O. K., Jr., "Development of Factual Warrants for Left Turn Channelization Through Digital Simulation," Ph.D. Dissertation, Texas A & M University , 1966.

Dawson, R. F. and Michael, H. L., "Analysis of On-Ramp Capacities by Monte Carlo Simulation," *Highway Research Record 118,* Highway Research Board, 1966, pp. 1-20.

DeCabooter, P. H. and Sinha, K. C., "Comparison Study by Computer Simulation of the Driver's Visual Part-Task During Left and Right Freeway Merging Maneuvers," *Highway Research Record 388,* Highway Research Board, 1972, pp. 1-12.

Dickey, J. W. and Montgomery, D. C., "A Simulation-Search Technique: An Example Application for Left-Turn Phasing," *Transportation Research,* Vol. 4, No. 4, 1970, pp. 339-347.

Diewald, W. J., "Investigation of a Combined Photographic and Computer Simulation Technique for Use in the Study of Isolated Intersections," Ph.D. Dissertation, Ohio State University, 1971.

Diewald, W. J. and Nemeth, Z. A., "Investigation of a Combined Photographic and Computer-Simulation Technique for Use in the Study of Isolated Intersections," *Highway Research Record 398,* Highway Research Board, 1972, pp. 12-14.

Douty, T. R., "GPSS and Traffic Flow Optimization," *Proceedings,* 8th Annual Conference and Movite Section Meeting, the University of Missouri Engineering Experiment Station, Series No. 65, November, 1966.

Eicher, J. P., "An Investigation of the Requirements for Collision Avoidance Systems at Railroad-Highway Grade Crossings--A Simulation Model," Ph.D. Dissertation, The Catholic University of America, 1970.

Fennessy, R. J. W., "A Computer Simulation Study of a Semi-Actuated Controlled Intersection," Research Report for Course 299, Civil Engineering Department, University of California, Berkeley, 1964.

Fisher, R. B., "Simulation of Traffic Flows at One-Way Restrictions," *Australian Road Research Board*, Vol. 2, Part 1, 1964, pp. 212-222.

Fox, P. and Lehman, F. G., "Computer Simulation of Single-Lane Automobile Traffic," Interim Report, Newark College of Engineering, Newark, N. J., October, 1965.

Fox, P. and Lehman, F. G., *Safety in Car Following--A Computer Simulation*, Newark College of Engineering, Newark, N. J., April, 1967.

Fox, P. and Lehman, F. G., "A Digital Simulation of Car Following and Overtaking," *Highway Research Record 199*, Highway Research Board, 1967, pp. 33-41.

Fox, P. and Lehman, F. G., "Digital Computer Simulation of Automobile Traffic," *Traffic Quarterly*, Vol. 21, No. 1, January, 1967, pp. 53-66.

Francis, J. G. F. and Lott, R. S., "A Simulation Programme for Linked Traffic Signals," paper presented at the Second International Symposium on the Theory of Traffic Flow, London, June 25-27, 1963.

Francki, M., "The Problem of Confidence and Three Methods of Variance Reduction in the Simulation of Queueing," *Australian Road Research*, Vol. 4, Part 1, 1968, pp. 567-581.

Gafarian, A. V., "Digital Computer Simulation of Diamond Interchanges," *Proceedings, Program Review Meeting, Research and Development of Traffic Systems*, Gaithersburg, Md., December, 1966, pp. 275-290.

Gafarian, A. V. and Walsh, J. E., "Statistical Approach for Validating Simulation Models by Comparison with Operational Systems--Illustrated for Traffic Flow," SP-2367, Systems Development Corporation, Santa Monica, California, 1966. Also in *Proceedings, Fourth International Conference on Operational Research*, 1966, pp. 702-705.

Gafarian, A. V., Hayes, E. and Mosher, W. W., Jr., "The Development and Validation of a Digital Simulation Model for Design of Freeway Diamond Interchanges," *Highway Research Record 208*, Highway Research Board, 1967, pp. 37-78.

Gafarian, A. V. and Walsh, J. E., "Methods for Statistical Validation of a Simulation Model for Freeway Traffic Near an On-Ramp," *Transportation Research,* Vol. 4, No. 4, 1970, pp. 379-384.

Gerlough, D. L., "Analogs and Simulation for the Study of Traffic Problems," *Proceedings Sixth California Street and Highway Conference,* 1954, pp. 82-83.

Gerlough, D. L., "Simulation of Freeway Traffic on a General-Purpose Discrete Variable Computer," Ph.D. Dissertation, University of California, Los Angeles, 1955.

Gerlough, D. L., Mathewson, J. H. and Trautman, D. L., "Study of Traffic Flow by Simulation," *Proceedings Highway Research Board 34,* 1955, pp. 522-530.

Gerlough, D. L., "Simulation of Freeway Traffic by an Electronic Computer," *Proceedings Highway Research Board 35,* 1956, pp. 543-547.

Gerlough, D. L., "Simulation of Freeway Traffic by Digital Computers," *Proceedings, Conference on Increasing Highway Engineering Productivity Held at Georgia Institute of Technology,* July 9-11, 1956, U. S. Bureau of Public Roads, 1957.

Gerlough, D. L., "A Comparison of Techniques for Simulating the Flow of Discrete Objects," paper presented at the National Simulation Conference, Dallas, Texas, October 23-25, 1958.

Gerlough, D. L., "Applications of Computers to Traffic Problems," *Proceedings, Institute of Traffic Engineers,* 1958, pp. 130-136.

Gerlough, D. L., "Traffic Inputs for Simulation on a Digital Computer," *Proceedings Highway Research Board 38,* 1959, pp. 480-492.

Gerlough, D. L. and Wagner, F. A., Jr., "Simulation of Traffic in a Large Network of Signalized Intersections," paper presented at the Second International Symposium on the Theory of Traffic Flow, London, June 25-27, 1963.

Gerlough, D. L., Wagner, F. A., Jr., Rudden, J. B., and Katz, J. H., "A Traffic Simulation Program for a Portion of the Traffic Signal System in the District of Columbia," Bureau of Public Roads, 1963.

Gerlough, D. L., and Wagner, F. A., "Simulation of the Network," *Proceedings, Conference on Traffic Surveillance, Simulation and Control,* Washington, D. C., Sept. 14-15, 1964, pp. 152-165.

Gerlough, D. L., "Simulation of Traffic Flow," in *Highway Research Board Special Report 79,* Highway Research Board, 1964.

Gerlough, D. L. and Wagner, F. A., "Improved Criteria for Designing and Timing Traffic Signal Systems," Final Report on NCHRP Project 3-5, Planning Research Corp., March, 1964.

Gerlough, D. L., "Simulation as a Tool in Traffic Control Systems Evaluation," in *Traffic Control Theory and Instrumentation,* T. R. Horton (editor), Plenum Press, New York, 1965.

Gerlough, D. L., and Wagner, F. A., "Improved Criteria for Traffic Signals at Individual Intersections," *National Cooperative Highway Research Program Report 32,* Highway Research Board, 1967.

Glickstein, A., Findley, L. D. and Levy, S. L., "A Study of the Application of Computer Simulation Techniques to Interchange Design Problems," Final Report for the U. S. Bureau of PUblic Roads, Midwest Research Institute, 1960. Also in *Highway Research Board, Bulletin 291,* 1961, pp. 139-162.

Glickstein, A. and Levy, S. L., "Application of Digital Simulation Techniques to Highway Design Problems," *Proceedings of the Western Joint Computer Conference,* Los Angeles, May 9-11, 1961, pp. 39-50.

Glickstein, A., "Analytic Methods in Transportation: Digital Simulation of Traffic," *Journal, Engineering Mechanics Division, Proceedings American Society of Civil Engineers,* Vol. 89, No. EM6, Part 1, 1963, pp. 1-13.

Goode, H. H., Pollmar, C. H. and Wright, J. B., "The Use of a Digital Computer to Model a Signalized Intersection," *Proceedings Highway Research Board 35,* 1956, pp. 548-557.

Goode, H. H., "The Application of a High Speed Computer to the Definition and Solution of the Vehicular Traffic Problem," *Operations Research,* Vol. 5, No. 6, 1957, pp. 775-793.

Goode, H. H., and True, W. C., "Simulation and Display of Four Inter-Related Vehicular Traffic Intersections," *13th National Meeting of the Association for Computing Machinery Proceedings,* 1958.

Grace, M. J., Morris, R. W. J., and Pak-Poy, P. G., "Some Aspects of Intersection Capacity and Traffic Signal Control by Computer Simulation," *Australian Road Research Board,* Vol. 2, Part 1, 1964, pp. 274-304.

Green, D. H., "Development of a Special Purpose Simulator for Detailed Traffic Studies," Ph.D. Dissertation, Victoria University, Manchester, England, 1963.

Green, D. H. and Hartley, M. G., "The Simulation of Some Simple Control Policies for a Signalized Intersection," *Operational Research Quarterly*, Vol. 17, No. 3, 1966, pp. 263-277.

Grimsdale, R. L., Mathers, R. W. and Sumner, F. H., "An Investigation of Computer-Controlled Traffic Simulation," *Proceedings Institute of Civil Engineers*, Vol. 25, Paper 6645, 1963, pp. 183-191.

Gross, V., Lattermann, D., Mertin, H. and Schnelle, N., "Simulation and Control of Road Traffic Networks," *Elektronische Rechenanlagen*, Vol. 7, No. 4, 1965, pp. 179-185.

Heathington, K. W., Miller, I., Knox, R. R., Hoff, G. C. and Bruggeman, J., "Computer Simulation of a Demand-Scheduled-Bus System Offering Door-to-Door Service," Civil Engineering Department, Northwestern University, Evanston, Ill., 1967.

Heathington, K. W. and Rath, G. J., "Computer Simulation for Transportation Problems," *Traffic Quarterly*, Vol. 22, No. 2, 1968, pp. 271-281.

Helly, W., "Dynamics of Single-Lane Vehicular Flow," Ph.D. Dissertation, M.I.T., 1959, reprinted as Research Report No. 2, Center for Operations Research, M.I.T., Cambridge, Mass.

Helly, W., "Simulation of Bottlenecks in Single-Lane Traffic Flow," paper presented at the First Symposium on the Theory of Traffic Flow, Warren, Michigan, 1959.

Hoffman, W. and Pavely, R., "Applications of Digital Computers to Problems in the Study of Vehicular Traffic," *Proceedings, Western Joint Computer Conference*, 1958, pp. 159-161.

Howat, M. G., "A Digital Computer Simulation of Driver Overtaking, Following, and Passing," *CAL Report No. VK-1938-V-I*, Cornell Aeronautical Lab., Inc., Buffalo, N. Y., July 15, 1965.

Huffman, R. A., and Walker, J. L., "Rapid Transit System Simulation with Interactive Graphics Display," *Transportation Planning and Technology*, Vol. 1, No. 3, 1973, pp. 205-217.

Jorgensen, N. O., "Determination of the Capacity of Road Intersections by Model Testing," *Ingenioren* (International Edition), Vol. 5, No. 3, 1961, pp. 99-101.

Katz, J. H., "Simulation of a Traffic Network," *Communications of the ACM*, Vol. 6, No. 8, 1963, pp. 480-486.

Kell, J. H., "Analyzing Vehicular Delay at Intersections Thorugh Simulation," *Highway Research Board, Bulletin 356*, 1962, pp. 28-39.

Kell, J. H., "Intersection Delay Obtained by Simulating Traffic on a Computer," *Highway Research Record 15,* Highway Research Board, 1963, pp. 73-97.

Kell, J. H., "Results of Computer Simulation Studies as Related to Traffic Signal Operation," *Proceedings, Institute of Traffic Engineers,* 1963, pp. 71-107.

Kell, J. H., "Simulation of the Intersection," *Proceedings, Conference on Traffic Surveillance, Simulation, and Control,* Washington, D. C., September 14-15, 1964, pp. 127-143.

Kell, J. H., "Traffic Simulation Validation," *Proceedings, Program Review Meeting, Research and Development of Traffic Systems,* Gaithersberg, Maryland, December, 1966, pp. 371-379.

Kell, J. H., "Intersection Simulation Model Validation," Final Report, Traffic Research Corporation, San Frncisco, 1966.

Kidd, E. A. and Laugher, K. R., "A Computer Model of Driving Behavior: The Highway Intersection Situation," CAL Report No. VJ-1843-V-I, Cornell Aeronautical Lab., Inc., Buffalo, N. Y., July 15, 1965.

Kidd, E. A. and Laughery, K. R., "Computer Model of Driver Behavior: The Highway Intersection Situation," *Highway Research Record 118,* Highway Research Board, 1966, pp. 96-97.

Kidd, E. A., and Laughery, K. R., "Urban Intersection: Defining Requirements for Crossing Maneuvers--Project Status," *Proceedings, Program Review Meeting, Research and Development of Traffic Systems,* Gaithersburg, Md., December, 1966, pp. 144-151.

Kobett, D. R., "A Digital Simulation Model of Freeway Traffic," Midwest Research Institute, Final Report MRI Project No. 2931-P, December, 1968.

Kobett, D. R., and Levy, S. L., "Study of Expressway Traffic Flow Through Digital Simulation," Final Report, Midwest Research Institute, Kansas City, Missouri, 1965.

Kreer, J. B. and Panyan, W. D., "Urban Traffic System Simulation and Control," Technical Report No. 11-3, Division of Engineering Research, Michigan State University , 1969.

Kroll, C. V., "Preview-Predictor Model of Driver Behavior in Emergency Situations," *Highway Research Record 364,* Highway Research Board, 1971, pp. 16-26.

Levy, S. L., Carter, M. C. and Glickstein, A., "Traffic and Simulation," paper presented at the Second International Symposium on the Theory of Road Traffic Flow, London, 1963.

Levy, S. L., "Simulation of the Freeway," *Proceedings, Conference on Traffic Surveillance, Simulation, and Control,* Washington, D. C., Sept. 14-15, 1964, pp. 166-174.

Lewis, R. M., "The Simulation of Vehicular Traffic at an Intersection on a Digital Electronic Computer," M. S. Thesis, Rensselear Polytechnic Institute, 1959.

Lewis, R. M., "The Simulation of Traffic Flow to Obtain Volume Warrants for Intersection Control," Ph.D. Dissertation, Purdue University, August, 1962. Reprinted as report for Joint Highway Research Project, No. 23, Purdue University, Lafayette, Indiana, 1962.

Lewis, R. M., "A Proposed Headway Distribution for Traffic Simulation Studies," *Traffic Engineering,* Vol. 33, No. 5, February, 1963, pp. 16-19, 48.

Lewis, R. M. and Michael, H. L., "Simulation of Traffic Flow to Obtain Volume Warrants for Intersection Control," *Highway Research Record 15,* Highway Research Board, 1963, pp. 1-43.

Lieberman, E. B., "Simulation of Traffic Flow at Signalized Intersections," General Applied Science Laboratories, Inc., 1967.

Lieberman, E. B., "Logical Design of the SCOT Model," KLD Assoc., Inc., Technical Report 2, 1971.

Lieberman, E. B., "Simulation of Corridor Traffic: the SCOT Model," *Highway Research Record 409,* Highway Research Board, 1972, pp. 34-45.

Lieberman, E. B., Worrall, R. D. and Bruggeman, J. M., "Logic Design and Demonstration of UTCS-1 Network Simulation Model," *Highway Research Record 409,* Highway Research Board, 1972, pp. 46-56.

Lindley, J. F., Nemeth, Z. and Reebel, J. O., "Digital Simulation of the Car-Following Situation," in *Study of Electronic Devices as Traffic Aids,* Report No. 202-1, Engineering Experiment Station, Ohio State University, Columbus, Ohio, 1962, pp. 5-18.

Lindley, J. F., Nemeth, Z. and Reebel, J. O., "Digital Simulation of Platoon Dynamics," in Study of Electronic Devices as Traffic Aids, Report No. 202-1, Engineering Experiment Station, Ohio State University, Columbus, Ohio, 1962, pp. 19-27.

Longley, D., "A Simulation Study of a Traffic Network Control Scheme," *Transportation Research,* Vol. 5, No. 1, 1971, pp. 39-57.

Manning, W. S., "A Procedure for Simulating Traffic Flow in a Network of Pretimed Signalized Intersections," M.S. Thesis, Virginia Polytechnic Institute, May, 1968.

Martin, W. A., "Description and Control of Single Lane Tunnel Traffic Flow," Research Report No. 3, Center for Operations Research, M.I.T., March, 1963.

May, A. D. and Pratt, D., "A Simulation Study of Load Factors at Signalized Intersections," *Traffic Engineering,* Vol. 38, No. 5, February, 1968, pp. 44-49.

May, A. D. and Gyamfi, P. "Extension and Preliminary Validation of a Simulation of Load Factor at Signalized Intersections," *Traffic Engineering,* Vol. 40, No. 1, October, 1969, pp. 46-52.

McHenry, R. R., "Computer Simulation of Automobile Accidents . . . A New Research Tool," *Simulation,* Vol. 11, No. 1, July, 1968, pp. 27-34.

Meeter, D. A., "A Simulation Program for a One-Lane, One-Way Highway System," FSU Statistics Report M133, February, 1968.

Miller, A. J., "A Computer Control System for Traffic Networks," paper presented at the SEcond International Symposium on the Theory of Road Traffic Flow, London, June 25-27, 1963.

Miller, E. T., "Investigation of Traffic Simulation Models for a Signalized Street Network," Ph.D. Dissertation, Texas A & M University, 1967.

Miller, S. B. and Little, J. D. C., "The Evaluation and Improvement of Traffic Signal Settings by Simulation," Tech. Report No. 22, Operations Research Center, M.I.T., Nov. 1966. Also published as *Highway Record 170,* Highway Research Board, 1967, pp. 56-69.

Montgomery, D. C., Talavage, J. J. and Mullen, C. J., "A Response Surface Approach to Improving Traffic Signal Settings in a Street Network," *Transportation Research,* Vol. 6, No. 1, 1972, pp. 69-80.

Morgan, H. L., "UTS-I: A Macro System for Traffic Network Simulation," *Proceedings, Spring Joint Computer Conference,* Atlantic City, 1970, pp. 217-222.

Morrison, J. W., Jr. and Moores, C. R., "The Application of Analog Computers to Traffic Intersection Problems," Arizone State University, December, 1962.

Naar, J., "Simulation of Vehicular Flow," M.S. Thesis, M.I.T., 1958.

Needler, M. A., "Computer Simulation for Traffic Control," *Proceedings, 56th Annual Road School,* Purdue University, 1970, pp. 70-78.

Olson, R. M., "Development of Break-Away Sign Support Structures," *Proceedings, Program Review Meeting, Research and Development of Traffic Systems,* Gaithersburg, Md., Dec., 1966, pp. 110-123.

Panyan, W. D., "Urban Traffic System Simulation and Control," Ph.D. Dissertation, Michigan State University, 1969.

Perchonok, P. A., "Application of Digital Simulation Techniques to Freeway On-Ramp Traffic Operations," MRK Project No. 2234-P, U. S. Bureau of Public Roads, June, 1954.

Perchonok, P. A. and Levy, S. L., "Application of Digital Computer Simulation Techniques to Freeway On-Ramp Traffic Operations," *Proceedings Highway Research Board 39,* Highway Research Board, 1960, pp. 506-523.

Petropoulos, D. P., "Simulation of Traffic Flow on One-Lane Roads with Turnouts," Ph.D. Dissertation, Stanford University, 1971.

Powner, E. T., "A New Traffic Simulator--An Aid to the Solution of Multi-Stream Flow Problems with Interference," U.T.S.G. Conference, Cranfield.

Pretty, R. L. and Blunden, W. R., "On the Computer Simulation of a Single Channel Queueing Facility for a Wide Range of Arrival and Departure Distributions," *Australian Road Research Board,* Vol. 2, Part 1, 1964, pp. 248-260.

Pretty, R. L., "The Effect of Right-Turning Vehicles on Saturation Flow Through Signalized Intersections," *Australian Road Research Board,* Vol. 3, Part 1, 1966, pp. 460-470.

Rhee, S. Y., "An Urban Traffic Control Simulator," paper presented at the Second International Symposium on the Theory of Traffic Flow, London, June 25-27, 1963.

Rhee, S. Y., "An Urban Traffic Control Simulator," M.S. Thesis, Case Institute of Technology, 1963.

Richard, R. M., Baker, R. L. and Sheldon, W. P., "Simulation of Traffic Flow as a Basis for Interchange Design," Final Report, Arizona Transportation and Traffic Institute, Tucson, 1965.

Romm, E. D., "Simulation of Traffic at a Two-Way Stop Intersection," School of Civil Engineering, Georgia Institute of Technology, Atlanta, Georgia, September, 1967.

Ruiter, E. R., "The Effects of Random Traffic on the Accuracy of the Vehicle Simulation and Operating Cost System," Research Report RC3-3, Department of Civil Engineering, Massachusetts Institute of Technology.

Ruiter, E. R. and Shuldiner, R. W., "Operating Costs at Intersections Obtained from the Simulation of Traffic Flow," U. S. Department of Commerce, Bureau of Public Roads, 1961.

Ruiter, E. R. and Shuldiner, P. W., "Operating Costs at Intersections Obtained from the Simulation of Traffic Flow," *Highway Research Record 89,* Highway Research Board, 1963, pp. 26-38.

Sagan, R., "Traffic Simulation with Cathode Ray Output," paper presented at the Fourth International Symposium on the Theory of Traffic Flow, Karlsruhe, 1968.

Sakai, T. and Nagao, M. "Simulation of Traffic Flow in a Network," *Communications of the ACM,* Vol. 12, No. 6, June, 1969, pp. 311-318.

Saleeb, S. I., "Logical Design and Associated Computer Studies for the Hardware Simulation of Traffic Dispersion," Ph.D. Dissertation, Manchester University, 1967.

Saleeb, S. I. and Hartley, M. G., "Simulation of Traffic Behavior Through a Linked-Pair of Intersections," *Transportation Research,* Vol. 2, No. 1, 1968, pp. 51-62.

Schalkwijk, W. F., "Simulation of Traffic Flow Through Large Traffic Nets," paper presented at the Fourth International Symposium on the Theory of Traffic Flow, Karlsruhe, 1968.

Schwartz, J. G., "An Urban Street Network Computer Simulation Model," Research Report R66-52, Civil Engineering Systems Laboratory, M.I.T., Cambridge, Mass., September, 1966.

Seddon, P. A., "A Program for Simulating the Dispersion of Platoons of Road Traffic," *Simulation,* Vol. 18, No. 3, 1972, pp. 81-90.

Shumate, R. P., and Dirksen, J. R., "A Simulation System for Study of Traffic Flow Behavior," *Highway Research Record 72,* Highway Research Board, 1965, pp. 19-39.

Silver, C. A., Farber, E., Weir, D. G. and McRuer, D. T., "Conceptualization of Overtaking and Passing on 2-Lane Rural Roads," Franklin Institute Research Lab. Technical Report 1-193, May, 1967.

Sinha, K. C., "The Development of a Digital Simulator for the Analysis of Freeway Traffic Phenomena," Research Report, Department of Civil Engineering, University of Connecticut, 1968.

Sinha, K. C. and Dawson, R. F., "Digital Computer Simulation of Freeway Traffic Flow," *Traffic Quarterly*, Vol. 24, No. 2, 1970, pp. 279-296.

Sinha, K. C. and DeCabooter, P. H., "A Computer Simulation Model of Driver Vision While Merging from a Freeway On-Ramp," *Traffic Quarterly*, Vol. 24, No. 4, 1972, pp. 589-613.

St. John, A. D., "Application and Expansion of a Traffic Simulation Model," *Proceedings, Program Review Meeting, Research and Development of Traffic Systems*, Gaithersburg, Md., Dec. 1966, pp. 159-172.

St. John, A. D., "A Study of Traffic Phenomena Through Digital Simulation," Final Report Research Grant AC 00106, MRI Project No. RA-13-P, Midwest Research Institute, 1967.

Stark, M. C., "Computer Simulation of Street Traffic," *NBS Tech. Note 129*, U. S. Department of Commerce, Off. Tech. Services, Washington, D. C.

Stark, M. C., "Computer Simulation of Nine Blocks of a City Street," *Highway Research Board, Bulletin 356*, 1962, pp. 40-47.

Stark, M. C., "Simulation of the Arterial Street," *Proceedings, Conference on Traffic Surveillance, Simulation and Control*, Washington, D. C., Sept. 14-15, 1964, pp. 144-151.

Story, C. E. R., "Simulation of Road Traffic Through a General Network," M.S. Thesis, University of Birmingham, England, 1969.

Story, C. E. R., "Simulation of Traffic by Digital Computer," *Traffic Engineering and Control*, Vol. 11, No. 10, February, 1970, pp. 464-467.

Sussam, E. D., Sugarman, R. C. and Knight, J. R., "Use of Simulation in a Study Investigating Alertness During Long-Distance, Low-Event Driving," *Highway Research Record 364*, Highway Research Board, 1971, pp. 27-32.

Taragin, A., "Simulation Needs in Integrated Traffic Control," *Proceedings, Conference on Traffic Surveillance, Simulation and Control*, Washington, D. C., September 14-15, 1964, pp. 175-182.

Testa, F. P. and Handelman, M., "Simulation of Garland, Texas, Vehicular Traffic Using Current and Computed Optimal Traffic Settings," paper presented at the 1973 Winter Simulation Conference, January, 1973.

Thomasson, J. N., Jr., "Simulation of Traffic at a Two-Way Stop Intersection," M.S. Thesis, Georgia Institute of Technology, Atlanta, 1966.

Thomasson, J. N., Jr. and Wright, P. H., "Simulation of Traffic at a Two-Way Stop Intersection," *Traffic Engineering*, Vol. 37, No. 11, August, 1967, pp. 39-45.

Tolle, J. E., "Digital Computer Simulation of the Car-Following Situation," a working paper on Project EES202 of the Transportation Engineering Center, Department of Civil Engineering, Ohio State University, Columbus, Ohio, 1965.

Trautman, D. L., Davis, H., Heilfron, J., Er-Chun-Ho and Rosenbloom, A., "Analysis and Simulation of Vehicular Traffic Flow," Institute of Transportation and Traffic Engineering, Research Report 20, University of California, Los Angeles, December, 1954.

Voskoglov, N. and Wheeler, P. J., "Optimizing a Local Street System by Simulation," *Traffic Quarterly*, Vol. 23, No. 2, 1969, pp. 179-196.

Wagner, F. A., Jr., "An Evaluation of Driver Decisions and Reactions at an Intersection Controlled by a Stop Sign," M.S. Thesis, Michigan State University, 1963.

Wagner, F. A., Rudden, J. B. and Gerlough, D. L., "Final Report, Study of the Traffic Signal System in a Portion of the District of Columbia, Volume I. Study Techniques and Results," Thompson Ramo Wooldridge, Inc., April, 1963.

Wagner, F. A., Barnes, F. C., Stirling, D. P. and Gerlough, D. L., "Urban Arterial and Network Simulation," Planning Research Corporation Report, Los Angeles, December, 1966.

Wagner, F. A., Barnes, F. C. and Gerlough, D. L., "Refinement and Testing of Urban Arterial and Network Simulation," Planning Research Corporation Report PRC-R-1064, Los Angeles, November, 1967.

Wagner, F. A. and Barnes, F. C., "Traffic Simulation Case Study: City of Detroit, New Center Area Network, "Planning Research Corporation Report PRC-R-1064B, Los Angeles, July 1968.

Wagner, F. A., Gerlough, D. L. and Barnes, F. C., "Improved Criteria for Designing and Timing Traffic Signal Systems: Urban Arterials," *National Cooperative Highway Research Program Report 73*, Highway Research Board, 1969.

Wagner, F. A., Barnes, F. C. and Gerlough, D. L., "Improved Criteria for Traffic Signal Systems in Urban Networks," *National Cooperative Highway Research Program Report 124*, Highway Research Board, 1971.

Walker, R. S. and Womack, B. F., "A Model for Traffic Simulation and Control," Technical Memorandum No. 15, Information Systems Research Laboratory, The University of Texas at Austin, January, 1970.

Walker, R. S., Womack, B. F. and Lee, C. E., "Traffic Network Simulation and Control with Driver Response Criteria," *Proceedings of the Houston Conference on Circuits, Systems and Computers*, April, 1970.

Walker, R. S., Womack, B. F. and Lee, C. E., "A Model for Traffic Simulation and a Simulation Language for the General Transportation Problem," *Proceedings, Fall Joint Computer Conference*, November 17-19, 1970, pp. 423-431.

Walton, J. R. and Douglas, R. A., "A LaGrangian Approach to Traffic Simulation on Digital Computers," *Highway Research Board, Bulletin 356*, 1962, pp. 48-50.

Warnshuis, P., "Simulation of Two-Way Traffic on an Isolated Two-Lane Road," *Transportation Research*, Vol. 1, No. 1, 1967, pp. 75-83.

Watjen, W. D., "Computer Simulation of Traffic Behavior Through Three Signals," *Traffic Engineering Control*, Vol. 6, No. 10, February, 1965, pp. 623-626.

Webster, F. V., "Traffic Signal Settings," Road Research Laboratory Technical Paper No. 39, HMSO, London, England, 1958.

Weir, D. H., "Simulation Studies of the Driver's Dynamic Response in Steering Control Tasks," *Highway Research Record 364*, Highway Research Board, 1971, pp. 1-15.

Wohl, M., "Simulation--Its Application to Traffic Engineering, Part I," *Traffic Engineering*, Vol. 30, No. 11, August, 1960, pp. 13-17, 29.

Wohl, M., "Simulation--Its Application to Traffic Engineering, Part II," *Traffic Engineering*, Vol. 31, No. 1, October, 1960, pp. 19-25, 56.

Wohl, M. and Brand, D., "Applications of Simulation to Highway Traffic Design," MIT Department of Civil Engineering, Report to U. S. Bureau of Public Roads, 1963.

Wong, S. R., "Traffic Simulator with a Digital Computer," *Proceedings, Western Joint Computer Conference*, San Francisco, California, February 7-9, 1956, pp. 92-94.

Wortham, A. W. and Baker, R. L., "A Macroscopic Event Scan Method of Simulating Traffic Flow in a Network," *Traffic Engineering*, Vol. 39, No. 2, November, 1968, pp. 42-45.

Wright, P. H., "Simulation of Traffic at a Four-Way Stop Intersection," Final Report, Project B-604, School of Civil Engineering, Georgia Institute of Technology, October, 1967.

Wright, P. H., Hassell, J. S., Jr., and Arrillaga, B., "Cross-Median Crashes," *Highway Research Report 332*, Highway Research Board, 1970, pp. 44-53.

Wright, P. H. and Arrillaga, B., "Simulation of Cross-Median Crashes," *Highway Research Record 388*, Highway Research Board, 1972, pp. 13-21.

VITA

Michael Pierce Deisenroth, son of Mr. and Mrs. Clifton E. Deisenroth, was born in Louisville, Kentucky, on April 8, 1944. He was graduated from Chattanooga High School in Chattanooga, Tennessee, in 1962.

He received the degree of Bachelor in Mechanical Engineering from Georgia Institute of Technology in June, 1967. As an undergraduate he received the Combustion Engineering Scholarship Award. During his undergraduate career, he was employed on the Cooperative Program by Sonoco Products Company of Hartsville, South Carolina.

Mr. Deisenroth entered the graduate program in the School of Industrial and Systems Engineering in September, 1967. He completed the requirements for the degree of Master of Science in Industrial Engineering in March, 1971. His thesis was entitled, "Quantitative Utilization of Activity Data for Initial Layouts."

During his graduate program, Mr. Deisenroth served as a teaching and research assistant. In June, 1971, he became an Instructor in the School of Industrial and Systems Engineering.

Mr. Deisenroth married the former Claudia Louise Mathews on June 18, 1966. Their first daughter, Margaret Elaine, was born May 15, 1969. Their second daughter, Michele Leigh, was born June 22, 1972.